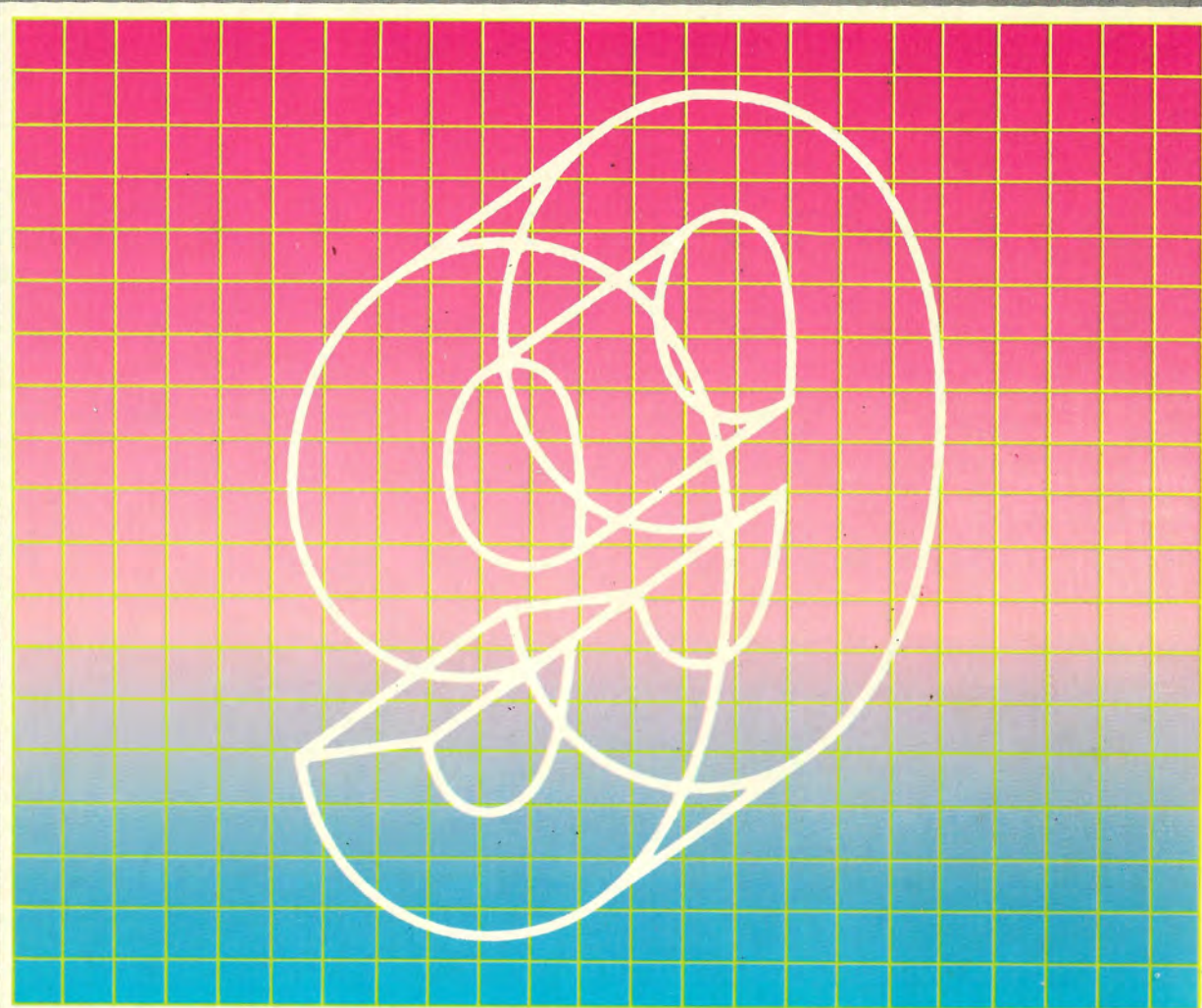


BIBLIOTECA PRACTICA

TALLER DE INFORMATICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

PROGRAMAS VALIDOS PARA AMSTRAD,
IBM, SPECTRUM, COMMODORE Y MSX



BRICOLAGE INFORMATICO

MANEJO DE ELEMENTOS GRAFICOS

EDICIONES SIGLO CULTURAL

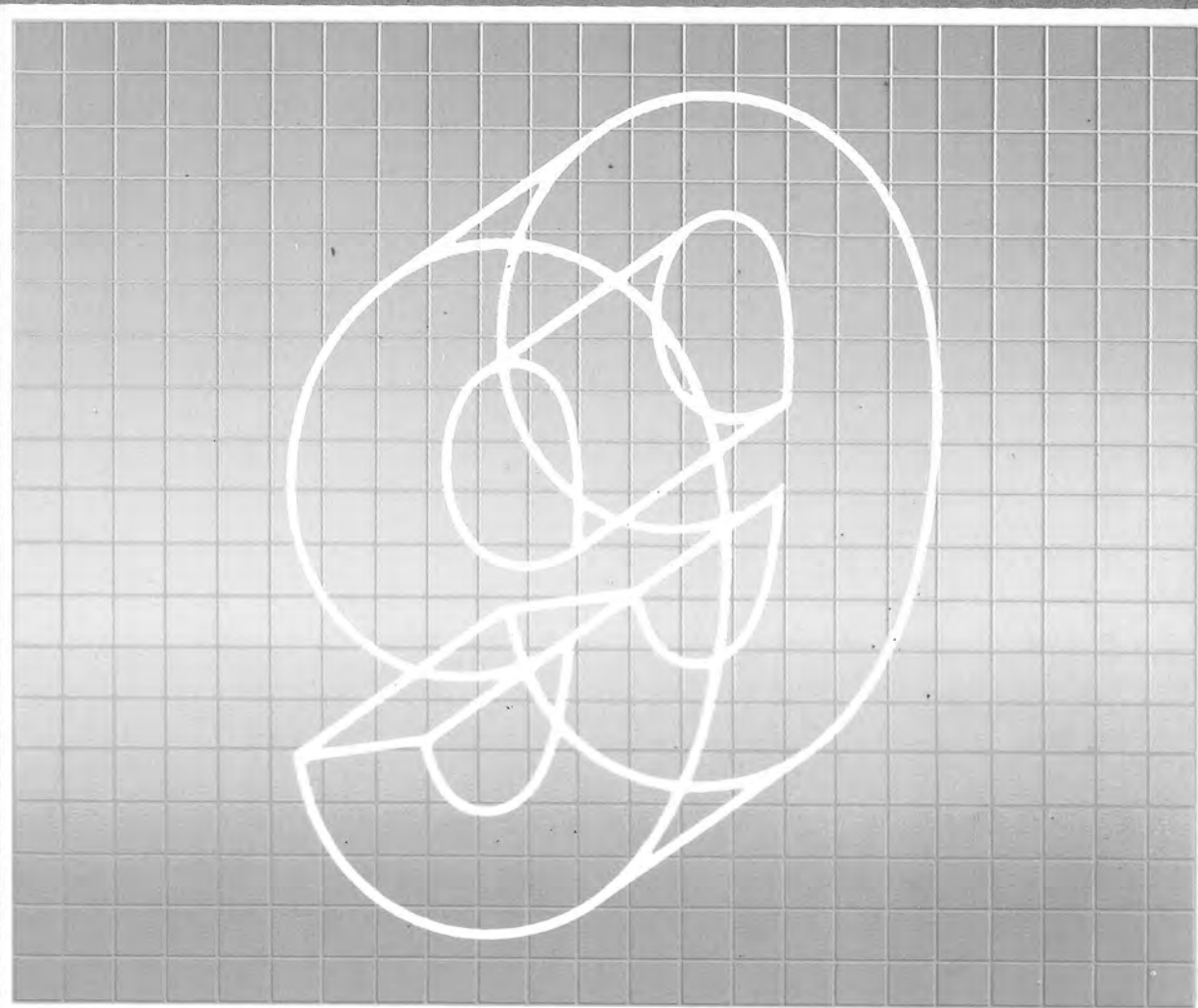
BIBLIOTECA PRACTICA

TRUCOS - SPRITES
BRICOLAGE DEL HARD
AULA ABIERTA
LOGO - TERMINOLOGIA

TALLER DE INFORMATICA

PROGRAMAS VALIDOS PARA AMSTRAD,

IBM, SPECTRUM, COMMODORE Y MSX



BRICOLAGE INFORMATICO

MANEJO DE ELEMENTOS GRAFICOS

EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S. A.

Director-editor:

RICARDO ESPAÑOL CRESPO.

Gerente:

ANTONIO G. CUERPO.

Directora de producción:

MARIA LUISA SUAREZ PEREZ.

Director de la colección:

JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño:

BRAVO-LOFISH.

Maquetación:

D. S. M.

Dibujos:

JOSE OCHOA

Tomo 9. «Experiencias prácticas en logo», Carlos Albert, Técnico de Informática; Adoración Llena Jubero, Técnico de Informática. «Manejo de sprites y elementos gráficos», «Trucos y rutinas básicas», Francisco Morales. Técnico de Informática. «Aprende con el ordenador», AULA DE INFORMATICA APLICADA: Soledad Tamariz-Martel, Diplomada en Telecomunicación; Francisco Blanca, Diplomado en Telecomunicación; Alejandro Marcos, Licenciado en Química; Jesús Bocho, Licenciado en Informática; Fernando Suero, Diplomado en Telecomunicación «El Taller del Hardware». Carlos Rey, Ingeniero Industrial. «Pequeña historia de la Informática», «Temas monográficos de vanguardia», «Terminología», «Vocabulario de Informática», Blanca Arbizu, Técnico de Informática.

Ediciones Siglo Cultural, S. A.

Dirección, redacción y administración:

Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:

Gofar Publicidad, S. A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:

COEDIS, S. A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.

Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:

CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.

Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-066-9

ISBN de la obra: 84-7688-047-2

Fotocomposición:

ARTECOMP, S. A. Albarracín, 50. 28037 Madrid.

Imprime:

MATEU CROMO. PINTO (Madrid).

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-43.185-1986

Printed in Spain - Impreso en España.

Suscripciones y números atrasados:

Ediciones Siglo Cultural, S. A.

Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid.

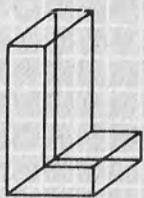
Marzo, 1987

INDICE

■ EXPERIENCIA Y PRACTICAS EN LOGO	5
■ MANEJO DE SPRITES Y ELEMENTOS GRAFICOS	15
■ TRUCOS Y RUTINAS BASICAS	25
■ EL TALLER DEL HARDWARE	33
■ APRENDER CON EL ORDENADOR	41
■ PEQUEÑA HISTORIA DE LA INFORMATICA	49
■ TEMAS MONOGRAFICOS DE VANGUARDIA	53
■ LOS LENGUAJES DE LA QUINTA GENERACION	57
■ VOCABULARIO DE INFORMATICA	59

EXPERIENCIA Y PRACTICAS EN LOGO

■ Procedimientos de entrada



OS procedimientos con entrada se diferencian de lo que hemos visto hasta ahora en que dependen de un valor o valores que nosotros le enviamos cuando lo llamamos (ejecutamos).

De momento sólo veremos los procedimientos con una entrada.

Pasamos el valor al procedimiento a través de una variable; esta variable tiene que acompañar al nombre del procedimiento cuando lo definimos.

```
? PARA CUADRADO :LADO
> REPITE 4 [ AV :LADO GD 90 ]
> FIN
```

Este procedimiento dibuja un cuadrado. Si queremos ejecutarlo y lo hacemos como hasta ahora lo hacíamos

```
? CUADRADO
```

el LOGO nos contestará que faltan datos para CUADRADO; efectivamente, ya que no le hemos dado ningún valor a la variable LADO.

En nuestro procedimiento el valor LADO define los puntos que debe avanzar la Tortuga (AV :LADO), y equivalen a la longitud de cada uno de los lados del cuadrado que dibujemos.

Para pasar el dato al procedimiento lo haremos así:

```
? CUADRADO 20
```

20 es el valor que toma la variable LADO.

Todas las variables que en un mismo procedimiento reciban el nombre LADO tomarán el valor 20, es decir, obtendríamos lo mismo que si hubiésemos definido CUADRADO de la siguiente forma:

```
? PARA CUADRADO
> REPITE 4 [ AV 20 GD 90 ]
> FIN
```

Con la diferencia que siempre que ejecutemos este último procedimiento obtendremos un cuadro de lado 20; en cambio, con el procedimiento con entrada podemos obtener, sin modificar nada, cuadrados de diferente lado; basta con ir pasando al procedimiento distintos valores de la variable.

```
? CUADRADO 20
```

Dibujaremos un cuadrado cuyo lado será 20.

```
? CUADRADO 35
```

el lado del cuadrado será 35.

No debemos olvidar nunca poner los dos puntos delante de la variable.

Te proponemos que dibujes dos veces dos cuadrados de diferente lado, uno encima de otro.

Cuando salgo del editor Logo, vuelvo al estado de la pantalla en que me encontraba, sin que nada de lo que había escrito o dibujado se haya borrado.

EXPERIENCIA Y PRACTICAS EN LOGO

(Sin borrar la pantalla entre ambos dibujos.)

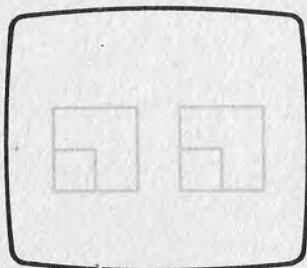


Fig. 1.

Con procedimientos sin entrada sería:

— Procedimiento para dibujar un cuadrado de lado 20:

```
? PARA CUADRADO1
> REPITE 4 [ AV 20 GD 90 ]
> FIN
```

— Procedimiento para dibujar un cuadrado mayor, de lado 40:

```
? PARA CUADRADO2
> REPITE 4 [ AV 40 GD 90 ]
> FIN
```

Procedimiento para inicializar la pantalla y para componer todo el dibujo:

```
? PARA DIBUJO
> BP
> SL
> PONPOS [ -50 -20 ]
> BL
> CUADRADO1
> CUADRADO2
> SL
> PONPOS [ 50 -20 ]
> BL
> CUADRADO1
> CUADRADO2
> FIN
```

Y lo ejecutaríamos:

```
? DIBUJO
```

Con procedimientos con entrada definiremos un solo procedimiento para dibujar los diferentes cuadrados.

— Procedimientos para dibujar cualquier tipo de cuadrado:

```
? PARA CUADRADO :LADO
> REPITE 4 [ AV :LADO GD 90 ]
> FIN
```

— Procedimiento para inicializar la pantalla y para componer todo el dibujo:

```
? PARA DIBUJAR
> BP
> SL
> PONPOS [ -50 -20 ]
> BL
> CUADRADO 20
> CUADRADO 40
> SL
> PONPOS [ 50 -20 ]
> SL
> CUADRADO 20
> CUADRADO 40
> FIN
```

Ejecución del dibujo:

```
? DIBUJAR
```

Hemos obtenido el mismo dibujo que con los procedimientos sin entrada, con la ventaja que nos hemos ahorrado un procedimiento.

A continuación vamos a definir un procedimiento con entrada que permita dibujar estrellas de diferentes tamaños.

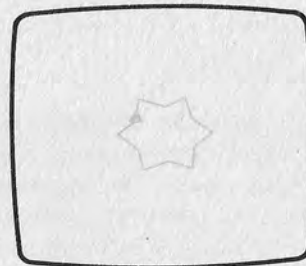


Fig. 2.

```
? PARA ESTRELLA :L
> REPITE 6 [ AV :L GD 100 AV :L GI 40 ]
> FIN
```

Los procedimientos con entrada utilizan la variable como dato.

Una vez tecleado el procedimiento podemos llamarle y darle diferentes valores a la variable L:

```
? BP ESTRELLA 10
? BP ESTRELLA 20
? BP ESTRELLA 30
```

También podemos dibujar unas encima de otras:

```
? PARA DIBUJO
> BP
> ESTRELLA 15
> PONRUMBO 180
> ESTRELLA 15
> PONRUMBO 90
> ESTRELLA 20
> PONRUMBO 270
> ESTRELLA 20
> FIN
```

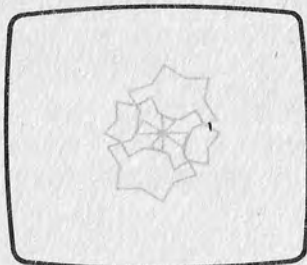


Fig. 3.

El siguiente ejemplo dibuja en la pantalla tres gráficas diferentes. La primera es una recta que parte del origen de los ejes y llega hasta el (50,50), indicando las coordenadas de distintos puntos de la recta.

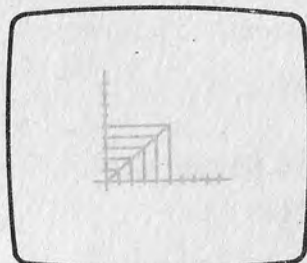


Fig. 4.

La segunda consiste en dibujar sobre los ejes anteriores una curva, indicando las coordenadas del punto final.

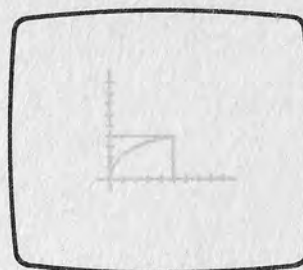


Fig. 5.

La tercera es un diagrama de barras.



Fig. 6.

Como para las tres gráficas utilizamos los mismos ejes, definimos un procedimiento para dibujarlos:

```
? PARA EJES
> BP
> PONCL 15
> SL
> PONPOS [ -100 -60 ]
> BL
> PONY 50
> SL
> PONPOS [ -110 -50 ]
> BL
> PONX 0
> SL PONPOS [ -90 -52 ]
> REPITE 2 [ REPITE 9 [ BL AV 4 RE 4
  GD 90 SL AV 10 GI 90 ] PON POS
  [ -98 -40 ] GI 90 ]
> FIN
```

A continuación definimos el que dibuja la recta:

```
? PARA RECTA
> EJES
> PONPOS [ -100 -50 ]
```

Si en los ordenadores que me lo permiten no especifico los procedimientos que quiero guardar, se guardarán todos los que haya definidos.

EXPERIENCIA Y PRACTICAS EN LOGO

```
> BL PONPOS [ -50 0 ]
> SL PONPOS [ -50 -51 ]
> GI 180
> PUNTOS 17
> SL PONPOS [ -60 -52 ]
> PUNTOS 14
> SL PONPOS [ -70 -50 ]
> PUNTOS 10
> SL PONPOS [ -80 -51 ]
> PUNTOS 7
> SL PONPOS [ -90 -49 ]
> PUNTOS 3
> FIN
```

En este procedimiento se llama a otro que recibe el nombre de PUNTOS, que es el que marca las coordenadas de diferentes puntos de la recta. Como la distancia de los ejes a los puntos de la recta es diferente en cada caso, utilizamos la variable L:

```
? PARA PUNTOS :L
> REPITE 2 [ REPITE :L [ BL AV 1 SL
AV 2 ] GI 90 ]
> GD 180
> FIN
```

El procedimiento para dibujar la curva es:

```
? PARA CURVA
> EJES
> PONPOS [ -100 -52 ]
> GI 180
> BL
> REPITE 9 [ GD 10 AV 8 ]
> SL PONPOS [ -50 -49 ]
> GI 90
> REPITE 13 [ BL AV 1 SL AV 2 ]
> BL
> GI 90
> REPITE 16 [ BL AV 1 SL AV 2 ]
> FIN
```

Para dibujar el diagrama de barras realizamos el siguiente procedimiento:

```
? PARA BARRAS
> EJES
> PONPOS [ -90 -49 ]
> GI 90
> PONCL 2
> BARRA 59
> SL PONPOS [ -70 -49 ]
> PONCL 5
```

```
> BARRA 49
> SL PONPOS [ -50 -49 ]
> PONCL 7
> BARRA 39
> SL PONPOS [ -30 -49 ]
> PONCL 13
> BARRA 29
> FIN
```

Utilizamos también en este caso otro procedimiento con entrada para dibujar cada barra, ya que cada una es longitud:

```
? PARA BARRA :LL
> BL
> REPITE :LL [ AV 10 RE 10 GI 90 AV
1 GD 90 ] AV 10
> FIN
```

La altura de la barra viene determinada por el valor de la variable LL.

Para obtener las tres gráficas seguidas permaneciendo cada una de ellas un tiempo determinado en la pantalla, definimos el siguiente procedimiento:

```
? PARA GRAFICAS
> PM
> BP
> OT
> RECTA
> ESPERA 100
> CURVA
> ESPERA 100
> BARRAS
> ESPERA 100
> FIN
```

Para ejecutarlo introduce la orden:

```
? GRAFICAS
```

■ Define tu propio juego de caracteres

En el tomo anterior definimos un juego de letras diferentes a las que normalmente salen en la pantalla. Estas eran más grandes y tenían una franja más ancha. Para ir completando el juego de caracteres, definimos en esta ocasión los números.

Los nombres que damos a los procedimientos corresponden al número que van a dibujar.

Podemos dar cualquier nombre a una variable.

Al igual que hacíamos con las letras, definimos al principio algunos procedimientos que se repiten en todos los demás.

Este procedimiento dibuja la franja ancha:

```
? PARA AA
> BL
> REPITE 3 [ AV 10 RE 10 GD 90
              AV 1 GI 90 ]
> FIN
```

En el siguiente posiciona a la Tortuga en la altura donde queremos dibujar los números. La situamos en este caso en la mitad de la pantalla ($Y=0$).

```
? PARA POSI
> SL
> PONY 0
> PONRUMBO 0
> FIN
```

Lo hemos fijado en 0, es decir, que dibuja los números en la posición central de la pantalla. Para variar esta posición, cambiar el valor de la orden PONY por la nueva posición en donde quieres dibujarlos:

```
? PARA CERO
> POSI
> AA
> AV 10 GI 90 AV 9
> GI 90 AV 10 GI 90
> AV 9 SL AV 10
> POSI
> FIN

? PARA UNO
> POSI
> AA
> GI 90 RE 2
> AV 9 RE 4 GD 90
> AV 10 GI 135 AV 4
> SL GI 135 AV 18
> POSI
> FIN
```

Si ahora introduces

```
? CERO UNO
```

aparecen los números cero y uno dibujados en la pantalla:

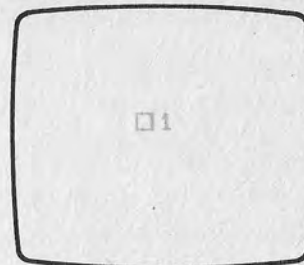


Fig. 7.

Continuamos definiendo los demás números:

```
? PARA DOS
> POSI
> SL AV 15 BL
> REPITE 2 [ AV 5 RE 5 GD 90 AV 1 GI
              90 ]
> AV 5 SL RE 10
> GI 90 BL AV 9
> GD 90 AV 5 GD 90
> AV 8 GI 90 AV 5
> GI 90 AV 8
> SL RE 18
> POSI
> FIN

? PARA TRES
> POSI
> AA
> REPITE 2 [ GI 90 AV 8 RE 8 GD 90
              AV 5 ]
> GI 90 AV 8
> SL RE 18
> POSI
> FIN

? PARA CUATRO
> POSI
> AA
> AV 10 RE 10 GI 90
> AV 2 GD 90 AV 5
> GI 90 AV 7 GD 90
> AV 5 SL GD 90
> AV 18
> POSI
> FIN

? PARA CINCO
> POSI
> BL
```

Indica que un procedimiento va a ser con entrada escribiendo el nombre de una variable al lado del nombre del procedimiento.

EXPERIENCIA Y PRACTICAS EN LOGO

```
> REPITE 2 [ AV 5 RE 5 GD 90 AV 1 GI 90 ]
> GI 90 AV 8 RE 8
> GD 90 AV 5 GI 90
> AV 8 GD 90 AV 5
> GD 90 AV 8
> SL AV 10
> POSI
> FIN
```

```
? PARA SEIS
> POSI
> BL
> REPITE 2 [ AV 5 RE 5 GD 90 AV 2 GI 90 ]
> GI 90 BL AV 8
> GD 90 AV 10 RE 5
> GD 90 AV 8 GD 90
> AV 5 SL
> GI 90 AV 10
> POSI
> FIN
```

```
? PARA SIETE
> POSI
> AA
> AV 10 GI 90 AV 8
> SL RE 18
> POSI
> FIN
```

```
? PARA OCHO
> POSI
> AA
> REPITE 2 [ REPITE 2 [ AV 5 GI 90 AV 9 GI 90 ] AV 5 ]
> SL GD 90 AV 11
> POSI
> FIN
```

```
? PARA NUEVE
> POSI
> AA
> AV 10 GI 90 AV 10
> GI 90 AV 5 GI 90
> AV 10 SL AV 12
> POSI
> FIN
```

Una vez que tenemos todos los números definidos, podemos escribir en la pantalla las cifras que queramos.

Si, por ejemplo, queremos que aparezca el número 1987 en la parte superior de la pantalla, tendremos que hacer lo siguiente:

Primero cambiamos en el procedimiento POSI el valor de la orden PONY. Edita el procedimiento y cambia el valor cero por el valor 60:

```
PONY 60
```

A continuación sal del editor y define el siguiente procedimiento:

```
? PARA CIFRA
> BP
> OT
> SL PONX 40
> UNO
> NUEVE
> OCHO
> SIETE
> FIN
```

Si ahora introduces:

```
? CIFRA
```

en la pantalla aparecerá la cifra que queramos dibujar.

A los usuarios del AMSTRAD

Hasta ahora no hemos hecho alusión al ordenador AMSTRAD en cuanto la Logo se refiere, ya que no posee un Logo de la potencia de otras versiones para otros ordenadores.

Este ordenador no posee un Logo que trabaje directamente en castellano, sino que dispone de un Logo en inglés al que posteriormente se le traduce las órdenes.

Veamos la forma de trabajar con el Logo en este ordenador, la forma de cargarlo en su versión inglesa, el proceso para realizar la traducción al castellano y la equivalencia de las primitivas vistas hasta ahora por otros ordenadores.

Podemos encontrar el Logo para el AMSTRAD en floppy. La forma de cargarlo es la siguiente:

— Una vez encendido el ordenador e introducido el floppy, teclea:

```
| CPM
```

a continuación pulsa RETURN.

No olvides que tienes que poner dos puntos delante de la variable.

— Cuando el signo de interrogación hace aparición en la pantalla, significa que el Logo está cargado, pero en inglés, y que está dispuesto para empezar a recibir órdenes.

— A continuación procedemos a realizar el proceso para traducir las órdenes, que ahora están en inglés, al castellano.

— Teclea e introduce la orden:

LOAD "CAST

En pocos segundos las órdenes estarán traducidas y todo queda dispuesto para comenzar a trabajar en castellano.

La equivalencia de algunas órdenes en el Logo del AMSTRAD son las siguientes:

- Avanza n Adelante n
- Retrocede n Atrás n
- Giraizquierda n Izquierda n
- Giraderecha n Derecha n
- Subelapiz Pluma_Invisible
- Bajalapiz Pluma_Invisible
- Ocultatortuga Esconder_tortuga
- MuestraTortuga ... Ver_tortuga
- Repite Repetir
- Azar n Aleatorio n "C

Esta orden genera un valor aleatorio y lo introduce en la variable C.

- Limpia Limpiar
- Goma Borrar

Para desactivar la orden BORRAR, hay que dar la orden PLUMA_VISIBLE.

- Borrarpantalla Empezar
- Pantallagrafica Pantalla_de_Graficos
- Pantallatexto Pantalla_de_texto
- Punto n m Punto_n_m

En cuanto al color:

- Poncl n Color_Tortuga n

Los colores de los que dispone son:

- 0 Color de fondo.
- 1 Color amarillo.
- 2 Color azul celeste.
- 3 Color rojo.

Con la orden VUELVE_TORTUGA la Tortuga aparece en el lado opuesto por el que salió.

A la hora de definir un procedimiento, tenemos que realizar los siguientes cambios:

- Sustituir la orden PARA por TO.
- Sustituir la orden FIN por END.

Un procedimiento que dibuje un triángulo equilátero hay que definirlo:

```
TO TRIANGULO
ADELANTE 30
DERECHA 120
ADELANTE 30
DERECHA 120
ADELANTE 30
END
```

Si queremos corregir un procedimiento lo primero que tenemos que hacer es editarlo, para lo que se da la orden:

EDITAR "nombre de procedimiento

Aparecen en la pantalla las órdenes que componen el procedimiento que hemos indicado. Con las teclas de movimiento del cursor nos desplazamos hasta el lugar donde queremos realizar algún cambio y lo efectuamos. Una vez modificado el procedimiento pulsamos la tecla COPY y la modificación realizada se tomará en cuenta, quedando el procedimiento actualizado.

Para obtener los nombres de todos los procedimientos que hay definidos damos la orden:

LISTAR

Para ver las órdenes que componen un procedimiento damos la orden:

LISTA PROCEDIMIENTO "nombre de procedimiento

Para borrar un procedimiento determinado:

ER "nombre de procedimiento

Consigue distintos tamaños de una misma figura utilizando procedimientos con entrada.

EXPERIENCIA Y PRACTICAS EN LOGO

Vamos a ver cómo se guardan y se cargan los procedimientos en este ordenador.

Si deseamos guardar los procedimientos que tenemos definidos debemos dar la orden:

SALVAR "nombre de fichero

Si lo que deseamos es cargar en la memoria del ordenador los procedimientos, utilizamos:

CARGAR "nombre de fichero

Para obtener una lista con todos los ficheros que existen en el floppy damos la orden:

DIRECTORIO

Para salir de Logo cuando hemos terminado de trabajar con él introducimos la orden:

ADIOS "ordenador

■ Las teclas y el editor II

Continuemos conociendo teclas cuyo uso nos facilita el trabajo y nos ahorra tiempo.

ORDENADORES MSX:

Las teclas que aparecen a continuación tienen su uso limitado para cuando estamos trabajando dentro del editor LOGO.

— INS: Introduce una línea vacía a partir de la posición en donde se encuentre el cursor.

— ESC y ↑: Pasa a la página anterior del editor.

— ESC y ↓: Pasa a la página siguiente del editor.

Llamamos página al conjunto de órdenes que pueden visualizarse a la vez en una misma pantalla.

— ESC ESC y ↑: Sitúa el cursor al principio del editor.

— ESC ESC y ↓: Sitúa el cursor al final del último texto que se encuentre en el editor.

— ESC ESC y →: Sitúa el cursor al final de texto que se encuentra en la pantalla.

— ESC ESC y ←: Sitúa el cursor al principio del texto que se encuentra en la pantalla.

También disponemos de estas dos funciones:

— ESC y ←: Sitúa el cursor al principio de la línea.

— ESC y →: Sitúa el cursor al final de la línea.

Estas dos funciones también las podemos obtener pulsando CTRL A y CTRL E, respectivamente.

ORDENADORES SPECTRUM

— E MODECAPS 6: Sitúa el cursor al final de la pantalla.

— E MODE CAPS 7: Sitúa el cursor al principio de la pantalla.

— E MODE E: Sitúa el cursor al final del texto.

— E MODE B: Sitúa el cursor al principio del texto.

— E MODE P: Sitúa el cursor en la página anterior.

— E MODE N: Sitúa el cursor en la página siguiente.

— SYS S: Anula el scrolling.

El efecto scrolling se produce cuando el texto que existe ocupa más de una página de pantalla.

ORDENADORES PC-COMPATIBLES:

— PgDn: Sitúa el cursor en la página siguiente, pudiendo visualizar las veinte líneas siguientes.

— PgUp: Sitúa el cursor en la página anterior, visualizando las veinte líneas anteriores.

— CTRL y PgDn: Sitúa el cursor al final del último texto de la página.

— CTRL y PgUp: Sitúa el cursor al principio del primer texto de la primera página.

— HOME: Sitúa el cursor al principio de la primera línea que se encuentra en la pantalla.

— END: Sitúa el cursor al final de la última línea que se encuentra en la pantalla.

Para ejecutar un procedimiento con entrada tienes que dar el nombre de éste, seguido del valor de la variable.

Ejercicios

1. Intenta realizar el siguiente dibujo utilizando los procedimientos de una entrada.

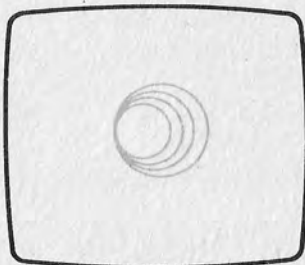


Fig. 8.

2. Define un procedimiento que sirva para dibujar cualquier triángulo equilátero y demuéstalo con un ejemplo.
3. Dibuja seis polígonos regulares de 4, 5, 6, 7, 8, 9 y 10 lados, respectivamente, con un solo procedimiento.
4. Intenta hacer el siguiente dibujo:

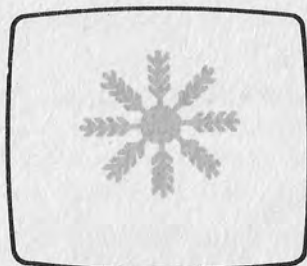


Fig. 9.

5. ¿Son correctas las siguientes órdenes?:

```
? PARA CIRCULO :20
....
....
> FIN
? PARA LINEA
> AV :LONG
> FIN
? PARA DIBUJO :LADO
> AV LADO
> FIN
```

Soluciones a los ejercicios

1:

Definimos el procedimiento que dibuja un círculo con un dato de entrada.

```
? PARA CIRCULO :CURVA
> REPITE 36 [ AV :CURVA GD 20 ]
> FIN
```

Y definimos un procedimiento para ir llamando al procedimiento CIRCULO asignando a la variable CURVA datos diferentes:

```
? PARA DIBU
> BP
> CIRCULO 4
> CIRCULO 5
> CIRCULO 6
> CIRCULO 7
> FIN
```

2:

```
? PARA TRIAN :LADO
> REPITE 3 [ AV :LADO GD 120 ]
> FIN
```

```
? PARA VARIOS
> BP
> TRIAN 20
> ESPERA 30
> BP
> TRIAN 40
> ESPERA 30
> BP
> TRIAN 60
> FIN
```

3:

```
? PARA FIGURA :N
> REPITE :N [ AV 30 GD 360 / :N ]
> FIN
```

```
? PARA DIBUJO
> BP
> GI 90
> FIGURA 4
> FIGURA 5
> FIGURA 6
> FIGURA 7
> FIGURA 8
> FIGURA 9
> FIN
```

Ahora un mismo procedimiento sirve para hacer diferentes cosas.

EXPERIENCIA Y PRACTICAS EN LOGO

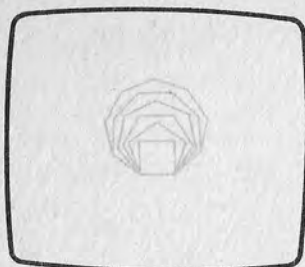


Fig. 10.

4:

Realizamos un procedimiento para dibujar una rama del cristal.

```
? PARA RAMA
> BL AV 10
> REPITE 4 [ 135 AV 10 GD 135 AV 5
              GD 45 AV 10 GI 45 AV 5 ]
> AV 5 GI 90 AV 5
> SL GD 90 RE 55
> GD 90 AV 4 BL
> AV 5 GI 90 AV 10
> REPITE 4 [ GD 135 AV 20 GI 135 AV
              5 GI 45 AV 10 GD 45
              AV 5 ]
> AV 5 GD 90 AV 5
> FIN
```

Ahora hacemos un procedimiento para situar las ramas en diferentes posiciones hasta completar el cristal:

```
? PARA CRISTAL
> PM BP OT
> SL PONPOS [ 0 -60 ]
> RAMA
> SL PONPOS [-51 -38 ]
> PONRUMBO 45
> RAMA
> SL PONPOS [ -71 14 ]
> PONRUMBO 90
> RAMA
> SL PONPOS [ -49 65 ]
> PONRUMBO 135
> RAMA
> SL PONPOS [ 3 85 ]
> PONRUMBO 180
> RAMA
```

```
> SL PONPOS [ 53 63 ]
> PONRUMBO -135
> RAMA
> SL PONPOS [ 74 12 ]
> PONRUMBO -90
> RAMA
> SL PONPOS [ 52 -39 ]
> PONRUMBO -45
> RAMA
> SL GI 90 AV 10
> BL RELLENA
> FIN
```

5:

```
? PARA CIRCULO :20
....
....
> FIN
```

Sí, es correcto.

:20 es el nombre de la variable a través de la cual pasamos los valores al procedimiento CIRCULO.

```
? PARA LINEA
> AV :LONG
> FIN
```

No es correcto.

La variable LONG no aparece detrás del nombre del procedimiento.

```
? PARA DIBUJO :LADO
> AV LADO
> FIN
```

No es correcto.

En la segunda línea faltan los dos puntos delante de la variable LADO.

```
? PARA :LADO
> AV 30
> FIN
```

Es correcto. El LOGO coge :LADO como nombre del procedimiento y no como variable, aunque lleve los dos puntos delante.

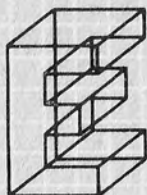
```
? PARA CUADRADO :FONDO
```

Es correcto. FONDO actúa como variable y tendrá un valor igual al código de color de fondo de la pantalla gráfica.

Podemos utilizar el mismo nombre de variable en diferentes procedimientos.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

Fantasma en tu ordenador



En el tomo anterior se propuso un programa que moviese un gusano por la pantalla. El movimiento habría de ser aleatorio. El movimiento podía variar en cualquier momento entre ser horizontal, vertical o diagonal.

Para ello se propuso utilizar la función RND, para elegir la dirección del movimiento de una forma aleatoria.

El programa 1, que aparece a continuación, nos muestra cómo se podía haber realizado este programa. En la figura 1 se dan las distintas posiciones que tendría el gusano según la dirección en la que se moviese.

<pre> ((((* O O O O O O ((((</pre> HORIZONTAL IZQUIERDA	<pre>)))) O O O O O O *)))) </pre> HORIZONTAL DERECHA
<pre> * O --O-- --O-- --O-- O </pre> VERTICAL ARRIBA	<pre> O --O-- --O-- --O-- O * </pre> VERTICAL ABAJO
<pre> O --O-- --O-- --O-- --O-- * </pre> DIAGONAL ++	<pre> * --O-- --O-- --O-- --O-- O </pre> DIAGONAL --
<pre> * --O-- --O-- --O-- --O-- O </pre> DIAGONAL +-	<pre> O --O-- --O-- --O-- --O-- * </pre> DIAGONAL -+

Fig. 1. Estas son las ocho definiciones del gusano que se mueve por la pantalla.

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```

10 REM *****
20 REM *** DEFINICION DEL GUSANO ***
30 REM *****
40 REM
50 REM *** MOV. HORIZONTAL IZQUIERDA ***
60 REM
70 LET A$(1)="      "
80 LET B$(1)="      "
90 LET C$(1)=" (((  "
100 LET D$(1)="*000000 "
110 LET E$(1)=" (((  "
120 LET F$(1)="      "
130 REM
140 REM *** MOV. HORIZONTAL DERECHA ***
150 REM
160 LET A$(2)="      "
170 LET B$(2)="      "
180 LET C$(2)=" )))) "
190 LET D$(2)=" 000000*"
200 LET E$(2)=" )))) "
210 LET F$(2)="      "
220 REM
230 REM *** MOV. VERTICAL ARRIBA ***
240 REM
250 LET A$(3)="  *  "
260 LET B$(3)="  O  "
270 LET C$(3)=" --O-- "
280 LET D$(3)=" --O-- "
290 LET E$(3)=" --O-- "
300 LET F$(3)="  O  "
310 REM
320 REM *** MOV. VERTICAL ABAJO ***
330 REM
340 LET A$(4)="  O  "
350 LET B$(4)=" --O-- "
360 LET C$(4)=" --O-- "
370 LET D$(4)=" --O-- "
380 LET E$(4)="  O  "
390 LET F$(4)="  *  "
400 REM
410 REM *** MOV. DIAGONAL ++ ***
420 REM
430 LET A$(5)="O      "
440 LET B$(5)="-O-     "
450 LET C$(5)=" -O-     "
460 LET D$(5)=" -O+     "
470 LET E$(5)=" -O-     "
480 LET F$(5)="  *      "
490 REM
500 REM *** MOV. DIAGONAL -- ***
510 REM
520 LET A$(6)="*      "
530 LET B$(6)="-O-     "
540 LET C$(6)=" -O-     "
550 LET D$(6)=" -O-     "
560 LET E$(6)=" -O-     "
570 LET F$(6)="  O      "
580 REM
590 REM *** MOV. DIAGONAL +- ***
600 REM
610 LET A$(7)="  *      "
620 LET B$(7)=" -O-     "
630 LET C$(7)=" -O-     "
640 LET D$(7)=" -O-     "
650 LET E$(7)=" -O-     "

```

```

660 LET F$(7)=" O "
670 REM
680 REM *** MOV. DIAGONAL -+ ***
690 REM
700 LET A$(8)=" O"
710 LET B$(8)=" -O-"
720 LET C$(8)=" -O- "
730 LET D$(8)=" -O- "
740 LET E$(8)=" -O- "
750 LET F$(8)=" * "
760 REM
770 REM *****
780 REM *** PROGRAMA GENERAL ***
790 REM *****
800 REM
810 REM *** MOVIMIENTO DEL GUSANO ***
820 REM
830 CLS
840 LET X=10
850 LET Y=10
860 GOSUB 890
870 ON DI GOSUB 990,1180,1370,1580,1790,2000,2210,2420
880 GOTO 860
890 REM
900 REM *** ELECCION DE LA DIRECCION ***
910 REM
920 IF X>=24 THEN LET DI=2:RETURN
930 IF X<=5 THEN LET DI=1:RETURN
940 IF Y>=15 THEN LET DI=3:RETURN
950 IF Y<=5 THEN LET DI=4:RETURN
960 LET DI=INT(RND*8+1)
970 RETURN
980 RETURN
990 REM
1000 REM *** MOVIMIENTO HACIA LA DERECHA ***
1010 REM
1020 FOR I=0 TO 3
1030 LOCATE Y,X+I
1040 PRINT A$(2)
1050 LOCATE Y+1,X+I
1060 PRINT B$(2)
1070 LOCATE Y+2,X+I
1080 PRINT C$(2)
1090 LOCATE Y+3,X+I
1100 PRINT D$(2)
1110 LOCATE Y+4,X+I
1120 PRINT E$(2)
1130 LOCATE Y+5,X+I
1140 PRINT F$(2)
1150 NEXT I
1160 LET X=X+I-1
1170 RETURN
1180 REM
1190 REM *** MOVIMIENTO HACIA LA IZQUIERDA ***
1200 REM
1210 FOR I=0 TO 3
1220 LOCATE Y,X-I
1230 PRINT A$(1)
1240 LOCATE Y+1,X-I
1250 PRINT B$(1)
1260 LOCATE Y+2,X-I
1270 PRINT C$(1)
1280 LOCATE Y+3,X-I
1290 PRINT D$(1)
1300 LOCATE Y+4,X-I
1310 PRINT E$(1)

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

```
1320 LOCATE Y+5,X-I
1330 PRINT F$(1)
1340 NEXT I
1350 LET X=X-I+1
1360 RETURN
1370 REM
1380 REM *** MOVIMIENTO HACIA ARRIBA ***
1390 REM
1400 FOR I=0 TO 3
1410 LOCATE Y-I,X
1420 PRINT A$(3)
1430 LOCATE Y-I+1,X
1440 PRINT B$(3)
1450 LOCATE Y-I+2,X
1460 PRINT C$(3)
1470 LOCATE Y-I+3,X
1480 PRINT D$(3)
1490 LOCATE Y-I+4,X
1500 PRINT E$(3)
1510 LOCATE Y-I+5,X
1520 PRINT F$(3)
1530 LOCATE Y-I+6,X
1540 PRINT " "
1550 NEXT I
1560 LET Y=Y-I+1
1570 RETURN
1580 REM
1590 REM *** MOVIMIENTO HACIA ABAJO ***
1600 REM
1610 FOR I=0 TO 3
1620 LOCATE Y+I,X
1630 PRINT " "
1640 LOCATE Y+I+1,X
1650 PRINT A$(4)
1660 LOCATE Y+I+2,X
1670 PRINT B$(4)
1680 LOCATE Y+I+3,X
1690 PRINT C$(4)
1700 LOCATE Y+I+4,X
1710 PRINT D$(4)
1720 LOCATE Y+I+5,X
1730 PRINT E$(4)
1740 LOCATE Y+I+6,X
1750 PRINT F$(4)
1760 NEXT I
1770 LET Y=Y+I-1
1780 RETURN
1790 REM
1800 REM *** MOVIMIENTO DIAGONAL ++ ***
1810 REM
1820 FOR I=0 TO 3
1830 LOCATE Y+I,X+I-2
1840 PRINT " "
1850 LOCATE Y+I+1,X+I
1860 PRINT A$(5)
1870 LOCATE Y+I+2,X+I
1880 PRINT B$(5)
1890 LOCATE Y+I+3,X+I
1900 PRINT C$(5)
1910 LOCATE Y+I+4,X+I
1920 PRINT D$(5)
1930 LOCATE Y+I+5,X+I
1940 PRINT E$(5)
1950 LOCATE Y+I+6,X+I
1960 PRINT F$(5)
```

```

1970 NEXT I
1980 LET X=X-I-1:LET Y=Y-I-1
1990 RETURN
2000 REM
2010 REM *** MOVIMIENTO DIAGONAL -- ***
2020 REM
2030 FOR I=0 TO 3
2040   LOCATE Y-I,X-I
2050   PRINT A$(6)
2060   LOCATE Y-I+1,X-I
2070   PRINT B$(6)
2080   LOCATE Y-I+2,X-I
2090   PRINT C$(6)
2100   LOCATE Y-I+3,X-I
2110   PRINT D$(6)
2120   LOCATE Y-I+4,X-I
2130   PRINT E$(6)
2140   LOCATE Y-I+5,X-I
2150   PRINT F$(6)
2160   LOCATE Y-I+6,X-I+1
2170   PRINT "      "
2180 NEXT I
2190 LET X=X-I+1:LET Y=Y-I+1
2200 RETURN
2210 REM
2220 REM *** MOVIMIENTO DIAGONAL +- ***
2230 REM
2240 FOR I=0 TO 3
2250   LOCATE Y-I,X+I
2260   PRINT A$(7)
2270   LOCATE Y-I+1,X+I
2280   PRINT B$(7)
2290   LOCATE Y-I+2,X+I
2300   PRINT C$(7)
2310   LOCATE Y-I+3,X+I
2320   PRINT D$(7)
2330   LOCATE Y-I+4,X+I
2340   PRINT E$(7)
2350   LOCATE Y-I+5,X+I
2360   PRINT F$(7)
2370   LOCATE Y-I+6,X+I-2
2380   PRINT "      "
2390 NEXT I
2400 LET X=X+I-1:LET Y=Y-I+1
2410 RETURN
2420 REM
2430 REM *** MOVIMIENTO VERTICAL -+ ***
2440 REM
2450 FOR I=0 TO 3
2460   LOCATE Y+I,X-I+2
2470   PRINT "      "
2480   LOCATE Y+I+1,X-I
2490   PRINT A$(8)
2500   LOCATE Y+I+2,X-I
2510   PRINT B$(8)
2520   LOCATE Y+I+3,X-I
2530   PRINT C$(8)
2540   LOCATE Y+I+4,X-I
2550   PRINT D$(8)
2560   LOCATE Y+I+5,X-I
2570   PRINT E$(8)
2580   LOCATE Y+I+6,X-I
2590   PRINT F$(8)
2600 NEXT I
2610 LET X=X-I+1:LET Y=Y+I-1
2620 RETURN

```

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

Como siempre, y antes de explicar el funcionamiento del programa, se dan todas las variaciones necesarias para que el programa funcione en ordenadores distintos del IBM pc, xt, at y compatibles.

COMMODORE:

```

68 DIM A$(8),B$(8),C$(8),D$(8),E$(8),F$(8)
830 PRINT CHR$(147)
960 LET DI=INT(RND(1)*8+1)
1030 POKE 214,Y:POKE 211,X+I
1050 POKE 214,Y+1:POKE 211,X+I
1070 POKE 214,Y+2:POKE 211,X+I
1090 POKE 214,Y+3:POKE 211,X+I
1110 POKE 214,Y+4:POKE 211,X+I
1130 POKE 214,Y+5:POKE 211,X+I
1220 POKE 214,Y:POKE 211,X-I
1240 POKE 214,Y+1:POKE 211,X-I
1260 POKE 214,Y+2:POKE 211,X-I
1280 POKE 214,Y+3:POKE 211,X-I
1300 POKE 214,Y+4:POKE 211,X-I
1320 POKE 214,Y+5:POKE 211,X-I
1410 POKE 214,Y-I:POKE 211,X
1430 POKE 214,Y-I+1:POKE 211,X
1450 POKE 214,Y-I+2:POKE 211,X
1470 POKE 214,Y-I+3:POKE 211,X
1490 POKE 214,Y-I+4:POKE 211,X
1510 POKE 214,Y-I+5:POKE 211,X
1530 POKE 214,Y-I+6:POKE 211,X
1620 POKE 214,Y+I:POKE 211,X
1640 POKE 214,Y+I+1:POKE 211,X
1660 POKE 214,Y+I+2:POKE 211,X
1680 POKE 214,Y+I+3:POKE 211,X
1700 POKE 214,Y+I+4:POKE 211,X
1720 POKE 214,Y+I+5:POKE 211,X
1740 POKE 214,Y+I+6:POKE 211,X
1830 POKE 214,Y+I:POKE 211,X+I-2
1850 POKE 214,Y+I+1:POKE 211,X+I
1870 POKE 214,Y+I+2:POKE 211,X+I
1890 POKE 214,Y+I+3:POKE 211,X+I
1910 POKE 214,Y+I+4:POKE 211,X+I
1930 POKE 214,Y+I+5:POKE 211,X+I
1950 POKE 214,Y+I+6:POKE 211,X+I
2040 POKE 214,Y-I:POKE 211,X-I
2060 POKE 214,Y-I+1:POKE 211,X-I
2080 POKE 214,Y-I+2:POKE 211,X-I
2100 POKE 214,Y-I+3:POKE 211,X-I
2120 POKE 214,Y-I+4:POKE 211,X-I
2140 POKE 214,Y-I+5:POKE 211,X-I
2160 POKE 214,Y-I+6:POKE 211,X-I+1
2250 POKE 214,Y-I:POKE 211,X+I
2270 POKE 214,Y-I+1:POKE 211,X+I
2290 POKE 214,Y-I+2:POKE 211,X+I
2310 POKE 214,Y-I+3:POKE 211,X+I
2330 POKE 214,Y-I+4:POKE 211,X+I
2350 POKE 214,Y-I+5:POKE 211,X+I
2370 POKE 214,Y-I+6:POKE 211,X+I-2
2460 POKE 214,Y+I:POKE 211,X-I
2480 POKE 214,Y+I+1:POKE 211,X-I
2500 POKE 214,Y+I+2:POKE 211,X-I
2520 POKE 214,Y+I+3:POKE 211,X-I
2540 POKE 214,Y+I+4:POKE 211,X-I
2560 POKE 214,Y+I+5:POKE 211,X-I
2580 POKE 214,Y+I+6:POKE 211,X-I

```

AMSTRAD Y MASX:

```

65 DIM A$(8),B$(8),C$(8),D$(8),E$(8),F$(8)
960 LET DI=INT(RND(1)*8+1)
1030 LOCATE X+I,Y
1050 LOCATE X+I,Y+1
1070 LOCATE X+I,Y+2
1090 LOCATE X+I,Y+3
1110 LOCATE X+I,Y+4
1130 LOCATE X+I,Y+5
1220 LOCATE X-I,Y
1240 LOCATE X-I,Y+1
1260 LOCATE X-I,Y+2
1280 LOCATE X-I,Y+3
1300 LOCATE X-I,Y+4
1320 LOCATE X-I,Y+5
1410 LOCATE X,Y-I
1430 LOCATE X,Y-I+1
1450 LOCATE X,Y-I+2
1470 LOCATE X,Y-I+3
1490 LOCATE X,Y-I+4
1510 LOCATE X,Y-I+5
1530 LOCATE X,Y-I+6
1620 LOCATE X,Y+I
1640 LOCATE X,Y+I+1
1660 LOCATE X,Y+I+2
1680 LOCATE X,Y+I+3
1700 LOCATE X,Y+I+4
1720 LOCATE X,Y+I+5
1740 LOCATE X,Y+I+6
1830 LOCATE X+I-2,Y+I
1850 LOCATE X+I,Y+I+1
1870 LOCATE X+I,Y+I+2
1890 LOCATE X+I,Y+I+3
1910 LOCATE X+I,Y+I+4
1930 LOCATE X+I,Y+I+5
1950 LOCATE X+I,Y+I+6
2040 LOCATE X-I,Y-I
2060 LOCATE X-I,Y-I+1
2080 LOCATE X-I,Y-I+2
2100 LOCATE X-I,Y-I+3
2120 LOCATE X-I,Y-I+4
2140 LOCATE X-I,Y-I+5
2160 LOCATE X-I+1,Y-I+6
2250 LOCATE X+I,Y-I
2270 LOCATE X+I,Y-I+1
2290 LOCATE X+I,Y-I+2
2310 LOCATE X+I,Y-I+3
2330 LOCATE X+I,Y-I+4
2350 LOCATE X+I,Y-I+5
2370 LOCATE X+I-2,Y-I+6
2460 LOCATE X-I+2,Y+I
2480 LOCATE X-I,Y+I+1
2500 LOCATE X-I,Y+I+2
2520 LOCATE X-I,Y+I+3
2540 LOCATE X-I,Y+I+4
2560 LOCATE X-I,Y+I+5
2580 LOCATE X-I,Y+I+6

```

SPECTRUM:

```

65 DIM A$(8,8):DIM B$(8,8):DIM
C$(8,8):DIM D$(8,8):DIM E$(8,8):DIM
F$(8,8)
950 IF Y=5 THEN LET DI=4:RETURN
1030 PRINT AT Y,X+I;
1050 PRINT AT Y+1,X+I;
1070 PRINT AT Y+2,X+I;
1090 PRINT AT Y+3,X+I;
1110 PRINT AT Y+4,X+I;

```

```

1130 PRINT AT Y+5,X+I;
1220 PRINT AT Y,X-I;
1240 PRINT AT Y+1,X-I;
1260 PRINT AT Y+2,X-I;
1280 PRINT AT Y+3,X-I;
1300 PRINT AT Y+4,X-I;
1320 PRINT AT Y+5,X-I;
1410 PRINT AT Y-I,X;
1430 PRINT AT Y-I+1,X;
1450 PRINT AT Y-I+2,X;
1470 PRINT AT Y-I+3,X;
1490 PRINT AT Y-I+4,X;
1510 PRINT AT Y-I+5,X;
1520 PRINT AT Y-I+6,X;
1620 PRINT AT Y+I,X;
1640 PRINT AT Y+I+1,X;
1660 PRINT AT Y+I+2,X;
1680 PRINT AT Y+I+3,X;
1700 PRINT AT Y+I+4,X;
1720 PRINT AT Y+I+5,X;
1740 PRINT AT Y+I+6,X;
1830 PRINT AT Y+I,X-I-2;
1850 PRINT AT Y+I+1,X+I;
1870 PRINT AT Y+I+2,X+I;
1890 PRINT AT Y+I+3,X+I;
1910 PRINT AT Y+I+4,X+I;
1930 PRINT AT Y+I+5,X+I;
1950 PRINT AT Y+I+6,X+I;
2040 PRINT AT Y-I,X-I;
2060 PRINT AT Y-I+1,X-I;
2080 PRINT AT Y-I+2,X-I;
2100 PRINT AT Y-I+3,X-I;
2120 PRINT AT Y-I+4,X-I;
2140 PRINT AT Y-I+5,X-I;
2160 PRINT AT Y-I+6,X-I+1;
2250 PRINT AT Y-I,X+I;
2270 PRINT AT Y-I+1,X+I;
2290 PRINT AT Y-I+2,X+I;
2310 PRINT AT Y-I+3,X+I;
2330 PRINT AT Y-I+4,X+I;
2350 PRINT AT Y-I+5,X+I;
2370 PRINT AT Y-I+6,X+I-2;
2460 PRINT AT Y+I,X-I-2;
2480 PRINT AT Y+I+1,X-I;
2500 PRINT AT Y+I+2,X-I;
2520 PRINT AT Y+I+3,X-I;
2540 PRINT AT Y+I+4,X-I;
2560 PRINT AT Y+I+5,X-I;
2580 PRINT AT Y+I+6,X-I;

```

Una vez vistas las modificaciones necesarias para que el programa nos pueda funcionar en ordenadores que no sean ni IBM ni compatibles, vamos a ver el funcionamiento del programa.

Para empezar hay que decir que el programa ha resultado así de largo porque se ha pretendido que sea lo más claro posible. Se podría haber hecho una rutina que sirviese para realizar los ocho tipos distintos de movimiento, pero se han escrito ocho rutinas distintas para que se pueda entender mejor el funcionamiento del programa.

Por otro lado, hay momentos en que el gusano va dejando rastros (líneas, círculos o

asteriscos) por la pantalla. Esto es así porque el programa no está del todo depurado. De todas maneras no importa, pues el programa no pretende ser un programa modelo, sino sólo ilustrar todo lo que hemos visto hasta ahora. Por ello se le propone al lector que, con los conocimientos que ha ido adquiriendo durante todos estos tomos y la práctica que haya podido coger realizando sus propios programas, intente resolver el problema de los rastros que deja el gusano y que intente construir una rutina que sirva para realizar los ocho movimientos posibles.

Una vez dicho esto, vamos a entrar de lleno en el programa y en su explicación.

Línea 65. Realmente esta línea sólo es necesaria ponerla en el SPECTRUM. Sirve para dimensionar los vectores que almacenarán las distintas figuras de que se compone el gusano según la dirección de movimiento.

Líneas 70 a 120. Se define la forma que tendrá el gusano cuando se mueva con movimiento horizontal de derecha a izquierda de la pantalla.

Líneas 160 a 210. Se define la forma del gusano cuando éste se mueve con movimiento horizontal de izquierda a derecha de la pantalla.

Líneas 250 a 300. Aquí definimos la figura que tendrá el gusano cuando se mueva por la pantalla con movimiento vertical de abajo hacia arriba.

Líneas 340 a 390. Definimos la forma del gusano cuando éste se mueva con movimiento vertical de arriba hacia abajo.

Líneas 430 a 480. Definimos el gusano que se moverá con movimiento diagonal Positivo-Positivo (++)).

Líneas 520 a 570. Aquí se define la forma del gusano cuando éste se mueva por la pantalla con movimiento diagonal Negativo-Negativo (--).

Líneas 610 a 660. Se define la forma que nos permitirá mover el gusano con movimiento Positivo-Negativo (+-).

Líneas 700 a 750. Definimos la última de las formas que tendrá el gusano. Esta corresponde al movimiento diagonal Negativo-Positivo (-+).

Estas primeras líneas del programa, como acabamos de ver, son las que se encargan de definir las diferentes formas que tendrá el gusano en la pantalla dependiendo del movimiento que esté realizando en cada momento. Como vemos en el programa, hacemos uso de seis vectores distintos. Cada tipo de movi-

MANEJO DE SPRITES Y ELEMENTOS GRAFICOS

miento está, por así decirlo, unido a uno de los elementos de cada vector. Así el movimiento horizontal de derecha a izquierda está relacionado con el elemento número dos de todos los vectores. El movimiento diagonal Positivo-Positivo está relacionado con el elemento número cinco. Esta coincidencia no es tal. Se realizó de esta manera para permitir la construcción de una sola rutina de movimiento, y para ahorrarnos el uso de las sentencias LOCATE (o PRINT AT en el SPECTRUM). Por ello se anima al lector que utiliza esta misma estructura para realizar la rutina que se proponía un poco más arriba.

En el siguiente grupo de líneas nos encontramos con el corazón del programa. Estas líneas son las encargadas de decidir en qué dirección se moverá el gusano por la pantalla. Estas líneas también regularán que no se salga de la pantalla.

Línea 830. Borrarnos la pantalla.

Línea 840. Se asigna a la variable numérica X el valor 10.

Línea 850. Se asigna a la variable numérica Y el valor 10. Estas dos últimas líneas nos sirven para colocar el gusano más o menos en el centro de la pantalla. Esta posición (10,10) será de la que parta el gusano cuando empiece a andar.

Línea 860. Hacemos una llamada a la rutina que se encuentra en la línea 890. Esta rutina es la encargada de comprobar si el gusano se va a salir de la pantalla, de impedirlo en caso de que ocurra y de decidir en qué dirección va a realizarse el siguiente movimiento.

Línea 870. Según el valor que tenga la variable numérica DI el control del programa se transferirá a una de las líneas que aparecen en la instrucción. Así si DI tiene valor uno (1), el control del programa irá a la línea 990. Si DI valiese cinco (5) el programa continuaría en la línea 1790. La variable DI nos la devuelve la rutina que se encuentra a partir de la línea 890 e indica la dirección que va a llevar el gusano un momento después.

Línea 880. Se hace un GOTO a la línea 860. Con esto se consigue que el gusano no pare de moverse, con lo que el programa no tiene fin. Para parar el programa habrá que pulsar la tecla BREAK o STOP (según de qué ordenador se trate).

El siguiente grupo de líneas son las encargadas de decidir la nueva dirección del gusano y de comprobar si éste se va a salir de la pantalla.

Línea 920. Se pregunta si la coordenada X de la posición del gusano es mayor o igual a 24. En caso afirmativo (estamos a punto de salirnos de la pantalla) se asigna a DI el valor dos (2). Esto sirve para que el siguiente movimiento sea horizontal de derecha a izquierda. Con ello se tiene la seguridad de que el gusano no va a salirse de la pantalla.

Línea 930. Se comprueba si X es menor o igual a cinco (5). En caso afirmativo (estamos tocando el borde izquierdo de la pantalla) se asigna a DI el valor 1. Con ello se consigue que la siguiente dirección sobre la que se moverá el gusano sea horizontal de izquierda a derecha.

Línea 940. Se comprueba que el gusano no está en el borde inferior de la pantalla. Si la variable numérica Y tiene un valor mayor o igual a 15, entonces cambiamos el movimiento del gusano y hacemos que se mueva con movimiento vertical de abajo hacia arriba. Con ello el gusano se separa del borde inferior de la pantalla.

Línea 950. Por último, se comprueba que el gusano no esté a punto de tocar el borde superior de la pantalla. Si Y tiene un valor menor o igual a cinco (5) es que el gusano está muy próximo a dicho borde. En ese caso hacemos que el movimiento vertical sea de arriba hacia abajo para separarnos del borde.

En este último grupo de líneas los límites de la pantalla son tan pequeños porque se ha pensado en las dimensiones de la pantalla en el SPECTRUM. Para los usuarios del IBM y del AMSTRAD se recomienda variar estas líneas por las que aparecen a continuación y poner la pantalla en el modo de 80 columnas por línea. Con ello se conseguirá que el gusano tenga más espacio para moverse.

```
920 IF X>=24 THEN LET DI=2:RETURN
```

```
940 IF Y>=15 THEN LET DI=3:RETURN
```

Los usuarios del MSX y del COMMODORE también pueden cambiar estas dos líneas por:

```
920 IF X>=32 THEN LET DI=2:RETURN
```

```
940 IF Y>=18 THEN LET DI=3:RETURN
```

Línea 960. En el caso de que el gusano no esté a punto de tocar ninguno de los bordes de la pantalla, se le pide al ordenador un número aleatorio entre 1 y 8 para elegir la nueva dirección del movimiento.

Línea 970. Una vez que sabemos en qué dirección se va a mover el gusano dentro de

unos instantes, devolvemos el control del programa a la parte central de éste.

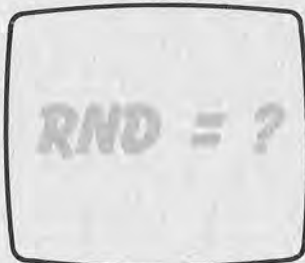


Fig. 2. Con el uso de la función RND, podemos hacer que la dirección del movimiento sea aleatoria.

RND = 1	→	MOVIMIENTO A LA DERECHA
RND = 2	→	MOVIMIENTO A LA IZQUIERDA
RND = 3	→	MOVIMIENTO HACIA ARRIBA
RND = 4	→	MOVIMIENTO HACIA ABAJO
RND = 5	→	MOVIMIENTO DIAGONAL ++
RND = 6	→	MOVIMIENTO DIAGONAL --
RND = 7	→	MOVIMIENTO DIAGONAL --
RND = 8	→	MOVIMIENTO DIAGONAL ++

Fig. 3. Dependiendo del número que nos de la función RND elegiremos la dirección y el sentido del movimiento del gusano.

Las siguientes líneas son las que se encargan de realizar cada uno de los movimientos que va a seguir el gusano. Estas líneas están agrupadas en ocho grupos, dependiendo de la dirección y del sentido del movimiento.

Líneas 1020 a 1170. Estas líneas son las encargadas de mover el gusano por la pantalla horizontalmente de izquierda a derecha. El bucle de la línea 1020 hace que dicho movimiento se repita cuatro veces. Si se quiere variar se puede hacer poniendo un número mayor o menor que tres después de la palabra TO. Hay que tener cuidado con dicho número, pues dentro de los bucles no se comprueba si el gusano se va a salir de la pantalla, por lo que si se pone un incremento demasiado grande puede hacer que el programa nos dé error. La explicación del bucle no se da porque, como ya hemos explicado este tipo de movimiento, el lector tiene que estar preparado para comprenderlo. La línea 1160 se encarga de ajustar el valor de la variable X después del movimiento.

Líneas 1210 a 1360. Son las que se encargan de mover el gusano horizontalmente

de derecha a izquierda. Como pasó en el anterior grupo de líneas, se puede variar la longitud del bucle que se encuentra en la línea 1210. Tampoco se explica cómo se realiza este movimiento, pues el lector ha de estar capacitado para comprenderlo.

Líneas 1410 a 1570. Esta rutina se encarga de mover el gusano por la pantalla con movimiento vertical de abajo hacia arriba.

Líneas 1610 a 1780. Se encargan de mover el gusano horizontalmente de arriba hacia abajo.

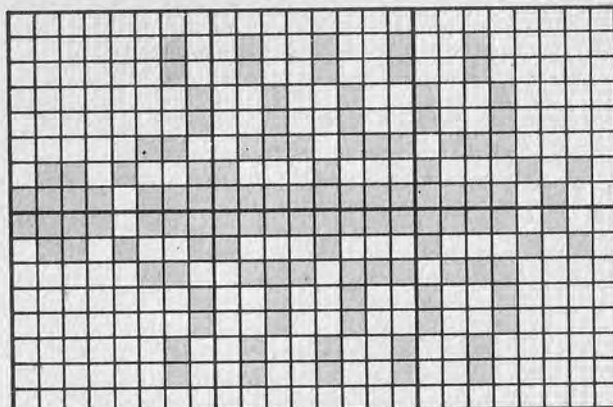
Líneas 1820 a 1990. Son las encargadas de mover el gusano con movimiento diagonal Positivo-Positivo (++). Aunque esta rutina (y las que la siguen) son algo más complicadas, el lector tiene que estar familiarizado con este tipo de rutinas.

Líneas 2030 a 2200. Se encargan de mover el gusano con movimiento diagonal Negativo-Negativo (--).

Líneas 2240 a 2410. Mueven el gusano con movimiento diagonal Positivo-Negativo (+-).

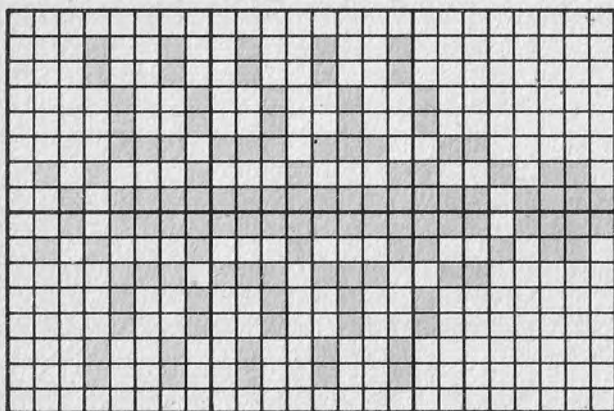
Líneas 2450 a 2620. Moverán el gusano por la pantalla con movimiento diagonal Negativo-Positivo (-+).

Como se puede apreciar al ejecutar el programa, éste no es muy vistoso, aunque ilustra cómo se pueden hacer programas que muevan figuras por la pantalla con cualquier tipo de movimiento. Como ejercicio, se propone al lector que, utilizando los caracteres definidos que aparecen en la figura 4 varíen este programa para que nos mueva un gusano por la pantalla, pero con más realismo. La solución a este pequeño problema no la daremos nosotros, por lo que es muy importante, antes de continuar con el control del movimiento, que cada uno intente hacer su propio programa para ir dejando los conceptos lo más claramente posible.

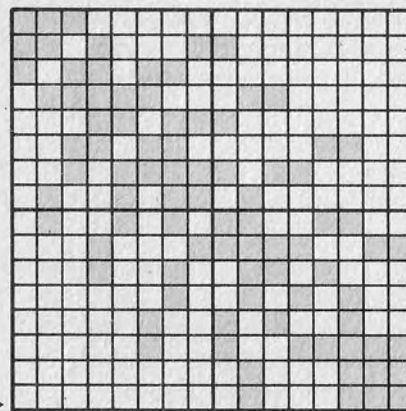


Horizontal Izquierda

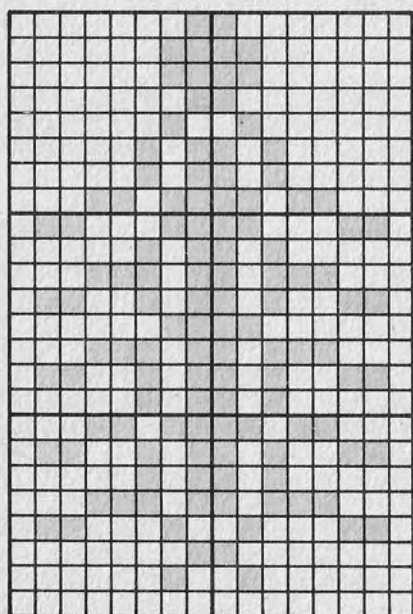
MANEJO DE SPRITES Y ELEMENTOS GRAFICOS



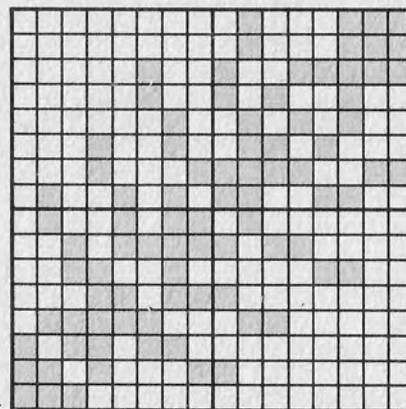
Horizontal
derecha



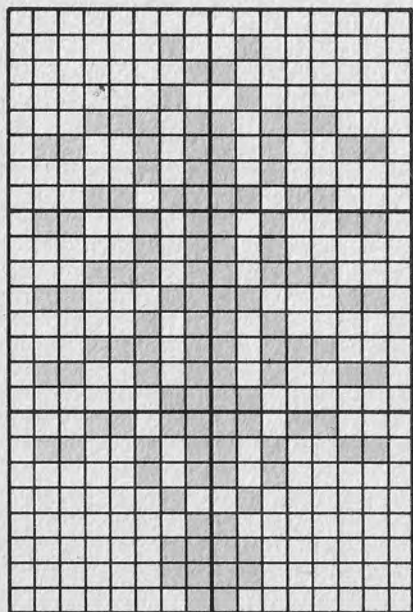
Diagonal
++



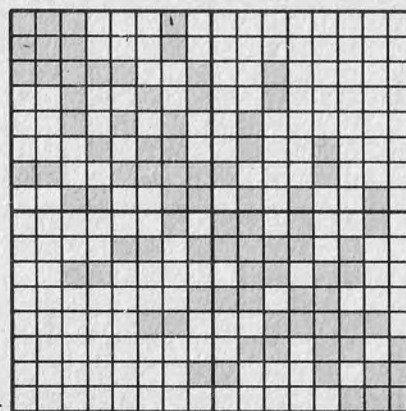
Vertical
arriba



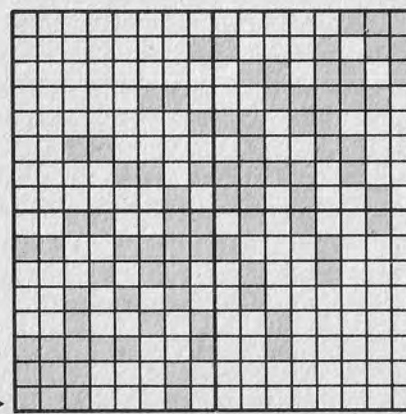
Diagonal
-+



Vertical
abajo

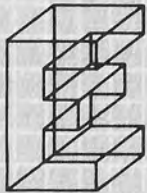


Diagonal
--



Diagonal
+-

TRUCOS Y RUTINAS BASICAS



STE tomo también lo dedicaremos al programa gestor de ficheros, del cual estamos hablando desde el tomo 6. Antes de empezar con la materia que compondrá este tomo vamos a recapitular sobre lo que hemos visto, y lo que no hemos visto, hasta ahora.

De momento hemos explicado cómo tiene que ser el programa que estamos realizando. Hemos dicho cuáles son las características que tiene que tener. Por otro lado, hemos ido viendo algunas de las rutinas que compondrán este programa. Estas rutinas, aunque algunas han sido construidas especialmente para funcionar con este programa, pueden ser utilizadas en cualquier programa que el lector crea conveniente. Las rutinas que hemos visto hasta ahora son:

- Rutina de datos de propósito general.
- Rutina de recogida de opción en el menú.
- Rutina de pulsa una tecla.

- Rutina de ordenación alfanumérica.
- Rutina de creación de menús.
- Rutina de búsqueda.

En este tomo veremos algunas rutinas más, pero no todas las que componen el programa de gestión de ficheros.

Antes de empezar con los programas pertenecientes a este tomo vamos a ver cómo haremos para enmascarar las fichas que queramos borrar de nuestro fichero. Como vimos en tomos anteriores, a la hora de borrar una ficha, ésta no se borrará de verdad, sino que le pondremos una marca para saber que ya no pertenece al fichero. Por otra parte, el programa nos debería de permitir, y de hecho lo hace, borrar de verdad todas aquellas fichas que han sido borradas, pero que realmente están enmascaradas. Aunque esto parezca una tontería no lo es. Esto nos puede servir para en cualquier momento recuperar una ficha que habíamos borrado. Si la ficha se hubiese borrado de verdad no habría forma de recuperarla, pero si sólo le ponemos una máscara entonces podemos ver su contenido en cualquier momento.

```
8500 REM
8501 REM *****
8502 REM * SUBROUTINA DE BUSQUEDA *
8503 REM *****
8504 REM
8505 CLS
8506 PRINT "B U S C A N D O"
8507 PRINT "=====
8508 PRINT
8509 IF SW=0 THEN PRINT "Buscando las fichas desde la ";N1;" hasta la ";N2
8510 IF SW=1 THEN PRINT "Buscando las fichas que tengan ";A$;" en el campo ";NC
```

TRUCOS Y RUTINAS BASICAS

```
8511 IF SW=2 THEN PRINT "Buscando fichas borradas"
8512 PRINT
8513 PRINT "ESPERA UN MOMENTO"
8514 FOR Z=1 TO 1000
8515 NEXT Z
8516 CLS
8517 ON SW+1 GOSUB 8526,8537,8567
8518 CLS
8519 PRINT "OPERACION TERMINADA"
8520 PRINT "=====
8521 PRINT
8522 PRINT "No se han encontrado mas fichas."
8523 LET X=1:LET Y=20
8524 GOSUB 8200
8525 RETURN
8526 REM
8527 REM *** BUSQUEDA POR No. DE FICHA ***
8528 REM
8529 FOR Z=N1 TO N2
8530     LET NF=Z
8531     GOSUB 9000
8532     LET X=1:LET Y=20
8533     GOSUB 8200
8534     CLS
8535 NEXT Z
8536 RETURN
8537 REM
8538 REM *** BUSQUEDA POR CAMPO Y VALOR ***
8539 REM
8540 LET CC=0
8541 FOR Z=1 TO TT
8542     IF MID$(F$(1),Z,1)=CHR$(254) THEN LET CC=CC+1
8543     IF CC=NC-1 THEN LET Z1=Z:LET Z=TT
8544 NEXT Z
8545 IF CC=NC-1 THEN GOTO 8554
8546 CLS
8547 PRINT "E R R O R"
8548 PRINT "-----"
8549 PRINT
8550 PRINT "Lo siento. El campo ";NC;" no existe."
8551 GOSUB 8200
8552 CLS
8553 RETURN
8554 LET C$=CHR$(254)
8555 FOR Z=1 TO TT
8556     FOR X=1 TO LEN(A$)
8557         IF MID$(F$(Z),X,1)<>MID$(A$,X,1) THEN LET X=LEN(A$)+100
8558     NEXT X
8559     IF X=LEN(A$)+101 THEN GOTO 8565
8560     LET NF=Z
8561     GOSUB 9000
8562     LET X=12:LET Y=20
8563     GOSUB 8200
8564     CLS
8565 NEXT Z
8566 RETURN
8567 REM
8568 REM *** BUSQUEDA DE FICHAS BORRADAS ***
8569 REM
8570 FOR Z=1 TO TT
8571     IF LEFT$(F$(Z),1)<>CHR$(253) THEN GOTO 8579
8572     LET F$(Z)=RIGHT$(F$(Z),LEN(F$(Z))-1)
8573     LET NF=Z
8574     GOSUB 9000
```

```

8575 LET X=1:LET Y=20
8576 GOSUB 8200
8577 CLS
8578 LET F$(Z)=CHR$(253)+F$(Z)
8579 NEXT Z
8580 RETURN

```

En el tomo anterior dejamos sin explicar la última parte del último programa porque no habíamos visto cómo íbamos a enmascarar las fichas borradas. Dicho programa lo volvemos a repetir aquí como el programa 1 y, después de explicar cómo enmascarar dichas fichas, vamos a ver cómo funciona la última parte de esta rutina.

Para enmascarar las fichas que deseamos borrar, lo único que tenemos que hacer es ponerle un carácter especial antes del contenido del primer campo. Esto es, tenemos que insertar al principio de la ficha un carácter que nos diga que ésta ya no existe. El carácter elegido es el que tiene código ASCII 253. Si te fijas en la figura 1, podrás ver qué carácter es éste en los distintos ordenadores.



Fig. 1. El carácter 253 en los distintos ordenadores.

La elección de este carácter no es arbitraria. Como vimos en tomos anteriores al especificar qué carácter actuaría como delimitador y separador de campos, es necesario elegir un carácter que no pueda ser introducido por el usuario desde el teclado. Los caracteres que tienen un código ASCII mayor de 127 no suelen ser fáciles de introducir por el teclado. Es por ello que hemos elegido el carácter 254 como delimitador de campos y el 253 (el anterior) como indicador de ficha borrada.

Una vez que sabemos cómo vamos a tratar e identificar a las fichas que, momentáneamente, queremos borrar vamos a ver el funcionamiento de las últimas líneas del programa 1.

Como vimos en el tomo anterior, este programa era el encargado de buscar una fi-

cha dentro del fichero. Como la búsqueda de las fichas puede hacerse siguiendo distintos parámetros, habíamos realizado un programa que nos permitía buscar una ficha bien por su número de orden, bien por el valor de un cierto campo. Como caso excepcional, el programa también nos permitía visualizar todas aquellas fichas que tuviesen la marca de *ficha borrada*. A continuación vamos a ver, línea a línea, la parte del programa que realiza esta última función:

Línea 8570. En esta línea comienza un bucle que va desde uno hasta el número máximo de fichas introducidas hasta el momento (*TT*), y dentro del cual iremos buscando aquellas fichas que tengan la marca de *borradas* e imprimiéndolas en pantalla.

Línea 8571. Se pregunta si el primer carácter de cada ficha, el que está más a la izquierda, tiene el código ASCII 253. En caso negativo (no es una ficha borrada) nos vamos a la línea 8579 para continuar con la siguiente ficha, para continuar con el bucle.

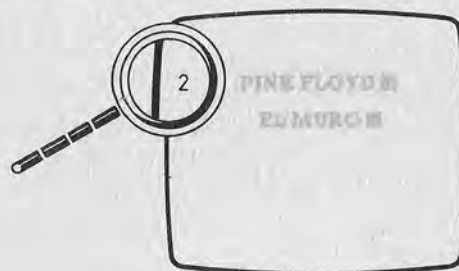


Fig. 2. Con sólo mirar el primer carácter de un registro podemos saber si dicha ficha está o no borrada.

Línea 8572. Si resulta que el primer carácter de la ficha que estamos examinando tiene código 253 entonces, momentáneamente, le quitamos dicho carácter. Con esto se consigue que dicha ficha no aparezca como borrada. Esto es necesario para poder imprimirla, ya que la rutina de impresión (que se encuentra a partir de la línea 9000) no imprime aquellas fichas cuyo primer carácter tenga el código ASCII igual a 253.

Línea 8573. Se asigna a la variable numérica *NF* el valor de *Z*. Como recordaremos,

TRUCOS Y RUTINAS BASICAS

a la rutina de impresión de fichas hay que pasarle el número de ficha que queremos imprimir. Este número hay que pasarlo en la variable *NF* (Número de Ficha). La ficha que estamos examinando ahora es la número *Z* (por el correr del bucle de la línea 8570).

Línea 8574. Llamamos a la rutina de impresión de fichas.

Línea 8575. Asignamos a la variable numérica *X* el valor uno (1) y a la variable *Y* el valor 20. Estas dos variables le dicen a la rutina que se encuentra a partir de la línea 8200 (Rutina de Pulsa una Tecla) en qué lugar tiene que aparecer el mensaje *PULSA UNA TECLA*.

Línea 8576. Hacemos una llamada a la rutina de pulsa una tecla. Cuando el usuario pulse cualquier tecla, se devolverá el control a la rutina que estamos estudiando.

Línea 8577. Borrarnos la pantalla.

Línea 8578. Como habíamos suprimido el carácter 253 de la ficha que acabamos de visualizar, y como dicha ficha es una ficha borrada, tenemos que volver a poner dicho carácter al principio de la ficha. Esto lo hacemos simplemente concatenando la ficha con el carácter 253.

Línea 7579. Aquí termina el bucle.

Línea 7580. Devolvemos el control del programa a la parte central de la rutina que compone el programa 1.

A continuación, y siguiendo con las fichas que en un cierto momento queremos borrar, vamos a ver la rutina que se encargará de borrar de verdad todas aquellas fichas que estaban pseudoborradas. El programa que realiza esto es el 2.

```
8300 REM
8301 REM *****
8302 REM * SUBROUTINA DE BORRADO REAL DE FICHAS *
8303 REM *****
8304 REM
8305 LET N=0
8306 FOR Z=1 TO TT
8307 IF LEFT$(F$(Z),1)<>CHR$(253) THEN GOTO 8313
8308 LET F$(Z)=""
8309 LET N=N+1
8310 FOR X=Z TO TT-1
8311 LET F$(X)=F$(X+1)
8312 NEXT X
8313 NEXT Z
8314 FOR Z=TT-N+1 TO TT
8315 LET F$(Z)=""
8316 NEXT Z
8317 LET TT=TT-N
8318 RETURN
```

Antes de pasar a explicar esta nueva subrutina vamos a ver las modificaciones que hay que realizar para que pueda ser ejecutada en el SPECTRUM.

SPECTRUM: 8307 IF F\$(Z,1)<>CHR\$(253) THEN GOTO 8313

El funcionamiento del programa, línea a línea es el siguiente:

Línea 8305. Inicializamos la variable numérica *N* a cero (0). Esta variable será la que vaya contando el número de fichas que vamos borrando.

Línea 8306. Aquí comienza un bucle que va desde uno (1) hasta el número máximo de fichas introducidas hasta el momento (TT) y dentro del cual iremos eliminando aquellas fichas que tengan el carácter 253 como primer carácter de la cadena.

Línea 8307. Se pregunta si el primer carácter de la ficha a la cual está apuntando el puntero *Z* está borrada. En caso negativo se transfiere el programa a la línea 8313 para continuar con la siguiente ficha.

Línea 8308. Si resulta que dicha ficha sí está borrada, entonces se borra el contenido del elemento del vector que tiene como subíndice el puntero.

Línea 8309. El contador de fichas borradas *N* se incrementa en uno.

Línea 8310. Comenzamos un bucle dentro del cual se va a borrar realmente la ficha que está pseudoborrada. En el grupo de líneas que vienen a continuación lo que se va a hacer es correr el contenido del fichero, a partir de la posición de la ficha borrada, un lugar hacia la izquierda.

Línea 8311. En esta línea hacemos que las fichas que se encuentran a partir de la ficha que acabamos de borrar se corran un lugar hacia abajo. Esto es, si acabamos de borrar la ficha número 23 entonces la ficha número 24 se convertirá en la 23, la 25 en la 24, la 26 en la 25 y así sucesivamente hasta que se termine el fichero.



Fig. 3

Línea 8312. Aquí termina el bucle que se encarga de reestructurar el fichero.

Línea 8313. Aquí termina el bucle principal, el bucle que se encarga de apuntar una a una a todas las fichas del fichero.

Línea 8314. Una vez que se ha terminado el bucle principal tenemos que borrar de la memoria las fichas que han quedado en la cola del fichero. Esto es, si hemos borrado tres fichas por ejemplo, esto significa que, si teníamos 25 fichas, ahora tenemos 22 fichas. Pero realmente si decimos al ordenador que nos imprima la ficha 23 o la 24 o la 25, va a resultar que dichas fichas existen y que además coinciden, que son la misma ficha. Esto es debido al corrimiento del fichero hacia números más bajos. En esta línea comienza un bucle dentro del cual nos vamos a encargar de borrar las fichas sobrantes en el fichero.

Línea 8315. Borrarnos ficha a ficha las que nos han quedado en la cola del fichero. Esto bien podía no hacerse, pero es mejor saber que tenemos limpio el fichero en cualquier momento.

Línea 8316. Aquí termina el bucle que se encarga de limpiar la cola del fichero.

Línea 8317. Como hemos borrado algunas de las fichas (en el caso que las hubiese), tenemos que ajustar el puntero que nos indica el número de fichas que contiene el fichero. Lo único que tenemos que hacer es restar del número antiguo de fichas el número de estas que hemos borrado en las líneas anteriores.

Línea 8318. Se devuelve el control al programa principal.

Ya hemos visto cómo podemos borrar de verdad las fichas que en un cierto momento el usuario puede haber pseudoborrado. Para hacer esta operación (seudoborrar) no hace falta escribir una rutina, ya que es una operación muy sencilla y que puede estar en el programa principal. Lo único que hay que hacer para que una ficha desaparezca, aparentemente, del fichero es concatenar el carácter 253 con dicha ficha al principio de ésta. Por ello la parte del programa que se encargará de hacerlo se compone de una sola línea como la siguiente:

```
LET F$(N)=CHR$(253)+F$(N)
```

donde N es el número de ficha que queremos borrar.

Por otro lado, la parte del programa que se encargará de recuperar una ficha que, aparentemente, esté borrada, también se compone de una sola línea, con lo que es una tonte-

ría ponerla en una rutina aparte. Esta línea sería algo así como:

```
LEF F$(N)=RIGHT$(F$(N),LEN(F$(N))-1)
```

donde N es número de la ficha que queremos recuperar.

La siguiente rutina que vamos a ver será la encargada de definir el formato de impresión que desea el usuario. Mediante una serie de preguntas el ordenador almacenará los datos necesarios para, a la hora de imprimir, saber lo que tiene que imprimir y cómo hacerlo. El programa del cual estamos hablando es el número 3.



Fig. 4. Esta rutina nos permitirá elegir cómo queremos los programas del fichero.

```
8000 REM
8001 REM *****
8002 REM * DEFINICION DE IMPRESION *
8003 REM *****
8004 REM
8005 CLS
8006 PRINT "DEFINICION DE LA IMPRESION"
8007 PRINT "===== "
8008 LOCATE 5,1
8009 PRINT "( CUANTOS CAMPOS QUIERE
      QUE IMPRIMA ?"
8010 LOCATE 12,1
8011 LINE INPUT "=> ";A$
8012 IF VAL(A$)=0 THEN RETURN
8013 LET N=VAL(A$)
8014 DIM N$(N,4)
8015 LOCATE 5,1
8016 PRINT "( QUE CAMPOS DESEA QUE
      IMPRIMA ? "
8017 LOCATE 9,1
8018 PRINT "CAMPOS A IMPRIMIR = "
8019 FOR Z=1 TO N
8020   LOCATE 12,1
8021   PRINT SPACE$(10)
8022   LOCATE 12,1
8023   LINE INPUT "=> ";N$(Z,1)
8024   IF VAL(N$(Z,1))=0 THEN GOTO 8020
8025   LOCATE 9,22
8026   FOR X=1 TO Z
8027     PRINT N$(X,1);";";
8028   NEXT X
8029 NEXT Z
8030 PRINT CHR$(29);" "
```

TRUCOS Y RUTINAS BASICAS

```

8031 LOCATE 5,1
8032 PRINT "( EN QUE ORDEN LOS IMPRIMO ? "
8033 PRINT "INTRODUZCA EL NUMERO DEL CAMPO EN EL "
8034 PRINT "ORDEN EN QUE QUIERE QUE SE IMPRIMA. "
8035 LOCATE 10,1
8036 PRINT "ORDEN DE IMPRESION = "
8037 FOR Z=1 TO N
8038     LOCATE 12,1
8039     PRINT SPACE$(10)
8040     LOCATE 12,1
8041     LINE INPUT "=> ";N$(Z,2)
8042     IF VAL(N$(Z,2))=0 THEN GOTO 8038
8043     LOCATE 10,22
8044     FOR X=1 TO Z
8045         PRINT N$(X,2);",";
8046     NEXT X
8047 NEXT Z
8048 PRINT CHR$(29);" "
8049 LOCATE 12,1
8050 PRINT "":REM 10 ESPACIOS EN BLANCO
8051 LET NN=0
8052 FOR Z=1 TO N
8053     FOR X=1 TO N
8054         IF N$(Z,1)=N$(X,2) THEN LET NN=NN+1:LET X=N+1
8055     NEXT X
8056 NEXT Z
8057 IF NN=N THEN GOTO 8077
8058 CLS
8059 PRINT "CAMPOS A IMPRIMIR = ";
8060 FOR Z=1 TO N
8061     PRINT N$(Z,1);",";
8062 NEXT Z
8063 PRINT CHR$(29);" "
8064 PRINT
8065 PRINT
8066 PRINT "ORDEN DE IMPRESION = ";
8067 FOR Z=1 TO N
8068     PRINT N$(Z,2);",";
8069 NEXT Z
8070 PRINT CHR$(29);" "
8071 LOCATE 12,1
8072 PRINT "NO COINCIDEN LOS CAMPOS A IMPRIMIR"
8073 PRINT "CON EL ORDEN DE IMPRESION"
8074 LET X=1:LET Y=20
8075 GOSUB 8200
8076 GOTO 8000
8077 LOCATE 15,1
8078 PRINT "( ESTA DE ACUERDO ? (S/N)"
8079 LET A$=INKEY$
8080 IF A$="S" OR A$="s" THEN GOTO 8084
8081 IF A$<>"N" AND A$<>"n" THEN GOTO 8079
8082 ERASE N$
8083 GOTO 8000
8084 FOR Z=5 TO 20
8085     LOCATE Z,1
8086     PRINT SPACE$(40)
8087 NEXT Z
8088 LOCATE 5,1
8089 PRINT "INTRODUZCA AHORA EN QUE COLUMNA"
8090 PRINT "DE LA IMPRESORA SE HA DE IMPRI-"
8091 PRINT "MIR CADA CAMPO."
8092 FOR Z=1 TO N
8093     LOCATE 12,1
8094     PRINT "CAMPO No. ";N$(Z,1);" ";

```

```

8095 LINE INPUT "=> ";N$(Z,3)
8096 IF VAL(N$(Z,3))=0 THEN GOTO 8093
8097 NEXT Z
8098 FOR Z=5 TO 17
8099 LOCATE Z,1
8100 PRINT SPACE$(40)
8101 NEXT Z
8102 LOCATE 5,1
8103 FOR Z=1 TO N
8104 PRINT "EL CAMPO ";N$(Z,1);" SE TABULARA EN LA COLUMNA ";N$(Z,3)
8105 NEXT Z
8106 LOCATE 20,1
8107 PRINT "( ESTA DE ACUERDO ? (S/N)"
8108 LET A$=INKEY$
8109 IF A$="S" OR A$="s" THEN GOTO 8112
8110 IF A$<>"N" AND A$<>"n" THEN GOTO 8108
8111 GOTO 8084
8112 FOR Z=5 TO 20
8113 LOCATE Z,1
8114 PRINT SPACE$(40)
8115 NEXT Z
8116 LOCATE 5,1
8117 PRINT "( CUANTAS LINEAS DEJO ENTRE FICHA"
8118 PRINT "Y FICHA? PULSA -1 PARA CAMBIAR DE"
8119 PRINT "PAGINA."
8120 LOCATE 12,1
8121 LINE INPUT "=> ";N$(1,4)
8122 IF VAL(N$(1,4))=0 THEN GOTO 8120
8123 CLS
8124 PRINT "OPERACION COMPLETADA"
8125 PRINT "=====
8126 LET X=1:LET Y=20
8127 GOSUB 8200
8128 CLS
8129 RETURN

```

Las modificaciones que hay que realizar para que esta rutina funcione en ordenadores distintos del IBM pc, xt, at y compatibles son las siguientes:

COMMODORE:

```

8005 PRINT CHR$(147)
8008 POKE 214,5:POKE 211,0
8010 POKE 214,12:POKE 211,0
8015 POKE 214,5:POKE 211,0
8017 POKE 214,9:POKE 211,0
8020 POKE 214,12:POKE 211,0
8021 FOR W=1 TO 10:PRINT " ";:NEXT W
8022 POKE 214,12:POKE 211,0
8025 POKE 214,9:POKE 211,21
8030 PRINT "<CURSOR-IZQ>"," "
8031 POKE 214,5:POKE 211,0
8035 POKE 214,10:POKE 211,0
8038 POKE 214,12:POKE 211,0
8039 FOR W=1 TO 10:PRINT " ";:NEXT W
8040 POKE 214,12:POKE 211,0
8043 POKE 214,10:POKE 211,21
8048 PRINT "<CURSOR-IZQ>"," "

```

```

8049 POKE 214,12:POKE 211,0
8058 PRINT CHR$(147)
8063 PRINT "<CURSOR-IZQ>"," "
8070 PRINT "<CURSOR-IZQ>"," "
8071 POKE 214,12:POKE 211,0
8077 POKE 214,15:POKE 211,0
8079 GET A$
8082 REM
8085 POKE 214,Z:POKE 211,0
8086 FOR W=1 TO 40:PRINT " ";:NEXT W
8088 POKE 214,5:POKE 211,0
8093 POKE 214,12:POKE 211,0
8099 POKE 214,Z:POKE 211,0
8100 FOR W=1 TO 40:PRINT " ";:NEXT W
8102 POKE 214,5:POKE 211,0
8106 POKE 214,20:POKE 211,0
8108 GET A$
8113 POKE 214,Z:POKE 211,0
8114 FOR W=1 TO 40:PRINT " ";:NEXT W
8116 POKE 214,5:POKE 211,0
8120 POKE 214,12:POKE 211,0
8123 PRINT CHR$(147)
8128 PRINT CHR$(147)

```

TRUCOS Y RUTINAS BASICAS

MSX Y AMSTRAD:

```
8008 LOCATE 1,5
8010 LOCATE 1,12
8015 LOCATE 1,5
8017 LOCATE 1,9
8020 LOCATE 1,12
8022 LOCATE 1,12
8025 LOCATE 22,9
8031 LOCATE 1,5
8035 LOCATE 1,10
8038 LOCATE 1,12
8040 LOCATE 1,12
8043 LOCATE 22,10
8049 LOCATE 1,12
8071 LOCATE 1,12
8077 LOCATE 1,15
8085 LOCATE 1,Z
8088 LOCATE 1,5
8093 LOCATE 1,12
8099 LOCATE 1,Z
8102 LOCATE 1,5
8106 LOCATE 1,20
8113 LOCATE 1,Z
8116 LOCATE 1,5
8120 LOCATE 1,12
```

SPECTRUM:

```
8008 PRINT AT 5,0;
8010 PRINT AT 12,0;
8011 INPUT LINE "=="> ";A$
8014 DIM N$(N,4,3)
8015 PRINT AT 5,0;
8017 PRINT AT 9,0;
8020 PRINT AT 12,0;
8021 FOR W=1 TO 10:PRINT " ";:NEXT W
8022 PRINT AT 12,0;
8023 INPUT LINE "=="> ";N$(Z,1)
8025 PRINT AT 9,21;
8030 REM
8031 PRINT AT 5,0;
8035 PRINT AT 10,0;
8038 PRINT AT 12,0;
8039 FOR W=1 TO 10:PRINT " ";:NEXT W
8040 PRINT AT 12,0;
8041 INPUT LINE "=="> ";N$(Z,2)
8043 PRINT AT 10,21;
8048 REM
8049 PRINT AT 12,0;
```

```
8070 REM
8071 PRINT AT 12,0;
8077 PRINT AT 15,0;
8082 REM
8085 PRINT AT Z,0;
8086 FOR W=1 TO 32:PRINT " ";:NEXT W
8088 PRINT AT 5,1;
8093 PRINT AT 12,0;
8095 INPUT LINE "=="> ";N$(Z,3)
8099 PRINT AT Z,0;
8100 FOR W=1 TO 32:PRINT " ";:NEXT W
8102 PRINT AT 5,0;
8106 PRINT AT 20,0;
8113 PRINT AT Z,0;
8114 FOR W=1 TO 32:PRINT " ";:NEXT W
8116 PRINT AT 5,0;
8120 PRINT AT 12,0;
8121 INPUT LINE "=="> ";N$(1,4)
```

Como puede apreciar el lector, el programa no reviste ninguna dificultad. Lo único que se hace es preguntar al usuario qué campos se quieren imprimir, en qué posición de la impresora y cuántas líneas de blanco hay que dejar entre ficha y ficha. Por supuesto, se pueden incluir muchísimas más preguntas, como, por ejemplo:

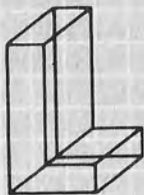
- Impresión de ciertos campos en negrilla o subrayados,
- Cuántas líneas en blanco entre los distintos campos.
- Cuál va a ser la cabecera (o pie) de cada página.

Pero estas preguntas obligan a trabajar con los caracteres de control de la impresora. Como éstos varían muchos de unas impresoras a otras, se deja al lector la puerta abierta para que el mismo efectúe los cambios necesarios en esta rutina.

El resultado de todas las preguntas que se le hacen al usuario queda almacenado en una matriz alfanumérica de $N \times 4$, donde N es el número de campos que se imprimen de cada ficha. Esta matriz será utilizada por la rutina de impresión cuando se la llame. También quedará almacenada en disco, o cassette, para que no se tenga que definir el formato de impresión cada vez que se carga el fichero.

EL TALLER DEL HARDWARE

Memorias



OS datos y los programas en los ordenadores se almacenan para su utilización en los circuitos que denominamos memoria. Todos los ordenadores poseen suficiente número de posiciones para poder realizar trabajos útiles o juegos de entretenimiento. En cualquier caso los circuitos utilizados son del mismo tipo: memorias de semiconductor. Vamos a ver con detenimiento las diferentes tecnologías de que pueden estar hechos, las señales que las controlan y las posibilidades de ampliación que existen en nuestros ordenadores personales.

Mostraremos algunos circuitos de aplicación que puedan conectarse a la tarjeta de ampliación de puertos o realizarse en una tarjeta específica de ampliación de memoria.

Tecnologías de memoria

Las memorias actualmente en uso en los ordenadores pueden dividirse según la tecnología con la que están fabricadas, de la manera siguiente:

- TTL. Rápidas, de poca densidad, para aplicaciones de almacenamiento transitorio en transferencias con dispositivos externos.

- IIL. De circuitos similares y de igual rapidez, pero utilizando transistores laterales, que necesitan menor espacio y también menor consumo.

- MOS en sus múltiples variantes: NMOS, HMOS y CMOS. Son más lentas, pero admiten mayor densidad de integración. Bajo consumo, sobre todo las CMOS.

- ECL. Muy rápidas, sólo utilizadas en los ordenadores de altas prestaciones o procesadores de señal.

- Burbujas. De uso limitado en aplicaciones que requieren seguridad ante fallos de alimentación. Utilizan la superficie magnética dividida en zonas mediante pequeñísimos circuitos que sirven de guía.

- Ferritas. Prácticamente en desuso, utilizan núcleos toroidales para el almacenamiento.

Por la forma como se realiza el acceso se pueden dividir en:

- De acceso al azar o directo (RAM). Diferentes organizaciones según el número de bits accedido simultáneamente: 1, 2, 4, 8 o 16 bits.

- De acceso secuencial. Registros de desplazamiento y acoplados por carga.

- De acceso por contenido, también llamadas asociativas (CAM=Content Addressable Memory).

Según el tipo de operación que pueda realizarse con la memoria podemos distinguir:

- De lectura y escritura (E/L o R/W).
- De sólo lectura.

El modo de empaquetamiento también permite distinguir dos tipos fundamentales en-

EL TALLER DEL HARDWARE

tre las de acceso al azar con operaciones de lectura y escritura:

- Estáticas
- Dinámicas

Estas clasificaciones nos permiten darnos una idea de la variedad de aplicaciones a las que se han adaptado los desarrollos tecnológicos para optimizar en cada caso la relación rendimiento/precio. Para las aplicaciones de nuestro interés relacionadas con nuestro ordenador personal es conveniente describir con cierto detalle la operación de algunos de los tipos mencionados para descubrir su utilidad.

Memorias RAM, bipolares

Una forma simple, aunque costosa, de realizar un elemento de memoria consiste en utilizar dos transistores multiemisor con las conexiones de la figura. Puede verse la forma como realizaríamos tanto la escritura como la lectura. Un pulso en la entrada A forzaría el circuito para que el transistor que esté conduciendo transmita el pulso a la línea de bit de salida. El transistor que esté al corte no propagará el pulso a su salida. La lectura de la línea activada se hace mediante amplificadores conectados a la línea de bit, que será común para todas las celdillas.

La lectura se realiza mediante la selección de la línea de palabra que se quiere leer, con decodificadores.

La escritura se realiza mediante otra entrada que cada transistor posee, activándose simultáneamente las dos entradas para hacer pasar a saturación el transistor que deba colocarse en estado de conducción, además de la entrada correspondiente de lectura/escritura.

El proceso de lectura/escritura es muy rápido, pero muy costoso en consumo de corriente, por lo que sólo se emplea en memo-

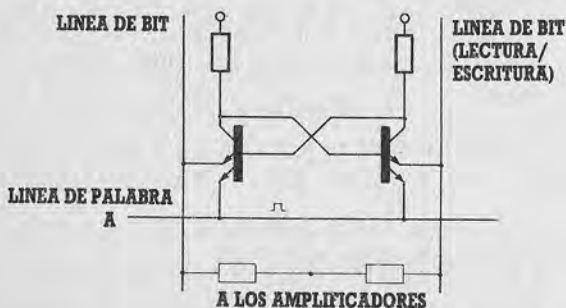


Fig. 1. RAM bipolar.

rias asociadas a la unidad central para acelerar en todo lo posible el proceso. También requieren mucho espacio físico. Los tiempos de acceso son de 25-50 nanosegundos.

Memorias RAM MOS, estáticas

La estructura básica es un biestable formado por 2 transistores acoplados entre sí por sus entradas y salidas cruzadas. La célula básica se compone además de otros dos transistores MOS que se utilizan como resistencias de carga y de otros 4 que se utilizan como puertas para la selección de lectura y grabación.

La lectura se efectúa introduciendo por las líneas X e Y pulsos positivos.

Por la línea de bit del transistor que esté conduciendo aparecerá el pulso, que será detectado por el amplificador correspondiente.

La grabación se efectúa inversamente, forzando la línea de bit con el nivel activo mientras que se generan los pulsos por las líneas X e Y. Si, por ejemplo, la línea de bit 1 se pone a la tensión de alimentación, el transistor T1 pasará a conducción, por lo que el transistor T2 pasará a corte y almacenará un 1.

Resultan menos voluminosas que las RAM bipolares y consumen menos potencia, pero siguen siendo más costosas que las dinámicas que veremos a continuación, pues requieren 8 transistores por célula. Los tiempos de acceso son del orden de 50-150 nanosegundos.

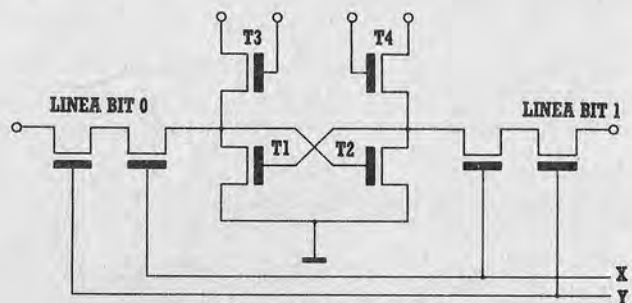


Fig. 2. RAM MOS estática.

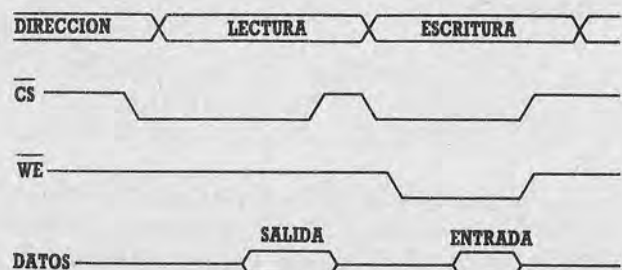


Fig. 3. Ciclo de lectura en RAM estática.

Memorias RAM MOS, dinámicas

Un fenómeno muy simple pero curioso ha permitido el abaratamiento y gran difusión de la memoria. Consiste en que trabajando con transistores MOS es posible "leer" la carga de un condensador sin que éste la pierda, o que lo haga muy lentamente. Basándose en este fenómeno se puede realizar una memoria que utilice un condensador como elemento de almacenamiento. Como la carga se pierde poco a poco es necesario disponer de un circuito que "refresque" el contenido de forma periódica, actualizando el valor que estuviera almacenado.

El esquema de la célula básica queda así reducido al de la figura, que resulta mucho más simple que el de su semejante estático. Por contra, el circuito de refresco y lectura resultan más complicados que el de las otras memorias.

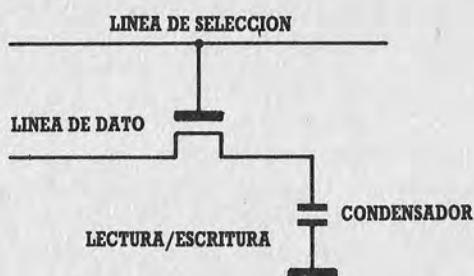
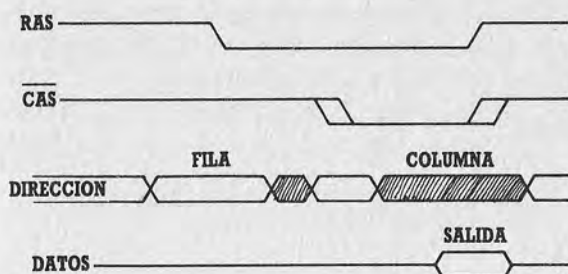


Fig. 4. RAM MOS dinámica.



No se muestra el circuito de refresco, que consiste en un amplificador que realimenta a la célula sobre la que se efectúa la lectura.

La forma práctica de presentarse es en circuitos que poseen un gran número de celdas, agrupadas en 1, 2, 4 u 8 bits por cada dirección. Una característica interesante y que ha permitido un alto grado de empaquetamiento es el multiplexado de los bits de dirección, con objeto de reducir el número de patañas necesarias para montar el chip y a la vez ser capaz de direccionar cada celda. Claro está que algo hay que pagar por la ventaja: se complica el circuito de control. Es necesario generar dos pulsos adicionales al de lectura o escritura: los pulsos necesarios para indicar la carga del registro interno de dirección en las dos fases. El primer pulso se denomina RAS (Row Address Strobe) porque indica que la dirección que hay en el bus corresponde a la fila, el segundo se denomina CAS (Column Address Strobe) para indicar que corresponde a la columna. Con la señal de selección de columna se activa un multiplexor, para dejar pasar la parte alta o la baja de la dirección a las entradas de los circuitos de memoria.

En la figura pueden verse los pulsos típicos para acceder a una memoria dinámica y los circuitos utilizados en los ordenadores personales para generarlos. Además es necesario generar un ciclo de refresco, para que la información se mantenga correctamente. El ciclo de refresco consiste básicamente en un

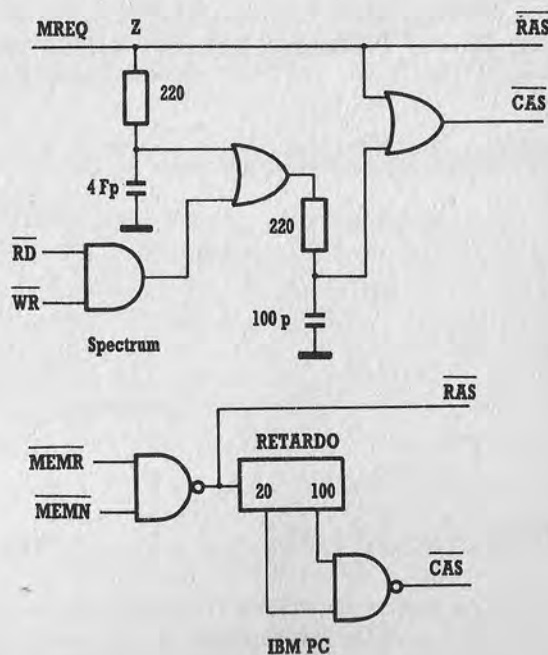


Fig. 5. Ciclo de lectura de RAM dinámica.

EL TALLER DEL HARDWARE

ciclo de lectura, pero que no llega a presentar los datos en el bus, por lo cual puede hacerse en todas las celdas de una fila a la vez. Para que la memoria sepa que el ciclo es de refresco los niveles de RAS y CAS son como los indicados en la figura.

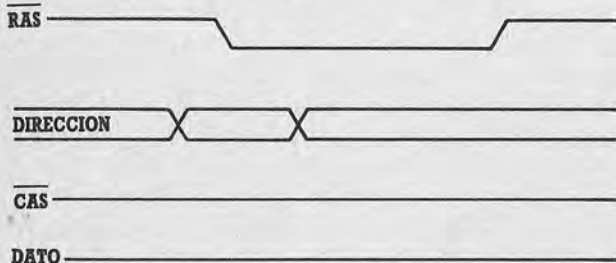


Fig. 6. Ciclo de refresco de RAM dinámica.

Memorias ROM

Para almacenamiento de programas o datos de forma permanente podemos disponer circuitos que mediante puentes indiquen el contenido de un bit. Así puede conseguirse que la presencia o ausencia del puente genere el pulso apropiado en el circuito de lectura. La programación de las memorias de sólo lectura puede hacerse mediante el diseño de la máscara que incluye los puentes o mediante fusibles programables por el usuario en una máquina apropiada.

Las memorias de sólo lectura programadas con máscara alcanzan una gran densidad y, por tanto, un precio muy económico, para almacenar programas permanentes de las máquinas, como pueden ser el soporte básico de los dispositivos o las rutinas del sistema operativo.

Memorias PROM

Las memorias que pueden ser grabadas por el usuario mediante una máquina apropiada se denominan PROM. Suelen disponer internamente de una red de fusibles que pueden direccionarse individualmente para su fundido. Mediante un programa adecuado se van fundiendo uno por uno los fusibles necesarios para que la PROM contenga el código que se desea.

Memorias EPROM

Para hacer el proceso de grabación reversible se han diseñado memorias en las cuales el elemento que almacena el contenido de

información es una celda especial con la puerta aislada. Este dispositivo es de funcionamiento muy interesante. El circuito del que se extrae la información es un transistor MOS, cuya puerta está muy próxima a la celdilla que almacena la información. Esta es en realidad la carga que se ha introducido y que permanece aislada, pues las paredes no son conductoras con las tensiones de trabajo inferiores a 5 voltios. Solamente mediante rayos ultravioleta las paredes se hacen ligeramente conductoras y, por tanto, se descarga cada celdilla hasta niveles de 0. El proceso de grabación requiere sobrepasar un cierto umbral para conseguir meter carga en la celdilla. Por eso los circuitos de memorias EPROM poseen una entrada para tensión más alta, capaz de introducir carga en las celdillas, cuando se realiza la conmutación adecuada.

El estado de fabricación y después del borrado de las EPROM es de todas las celdas en 1 lógico, normalmente. Disponen de una ventana de cuarzo transparente, para que pueda traspasar la luz ultravioleta hasta la ventana de cada celdilla.

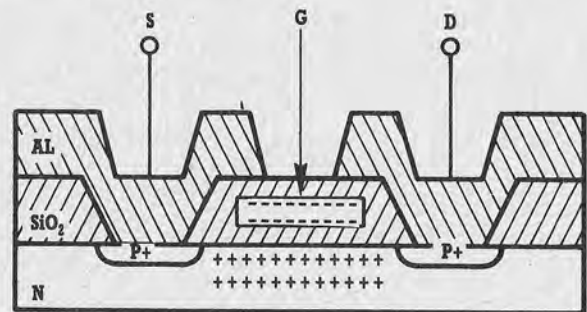


Fig. 7. Memoria EPROM.

Las señales necesarias para acceder a la información en estas memorias es prácticamente la misma que para las RAM estáticas, como se muestra en la figura.

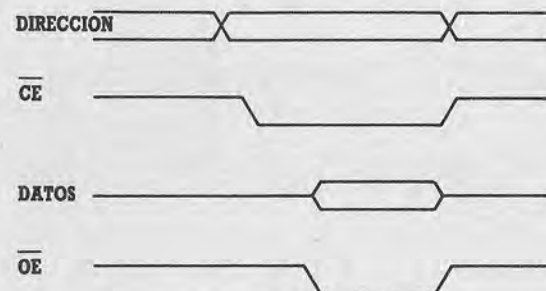


Fig. 8. Ciclo de acceso a EPROM.

Como ligera diferencia suelen disponer de una entrada para el control de los amplificadores de tres estados de la salida, para co-

nectarse al bus solamente en el momento en que la lectura ha terminado.

El proceso de grabación consiste en la presentación de los datos como si de un ciclo de escritura se tratase, pero activando a la vez una entrada específica con tensión más alta, que puede ser de 12 a 27 voltios, según el fabricante y el modelo. Durante el proceso de grabación puede realizarse un ciclo de lectura para verificar que se ha grabado la información deseada. En la figura pueden verse los pulsos correspondientes a grabación y verificación. Hay que resaltar que los tiempos de grabación son bastante críticos. Si se exceden las especificaciones puede sobrecalentarse la celdilla y hacerse irreversible el proceso de grabación/borrado.

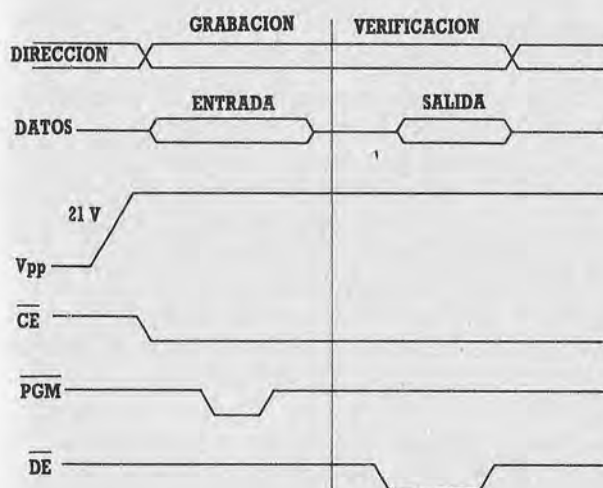


Fig. 9. Ciclo de grabación y verificación para 2764.

Las EPROM actuales se graban mediante pulsos de nivel lógico. Los modelos antiguos como la 2708 requerían pulsos de 25 voltios, lo que hacía ligeramente más complicada la grabación.

Las EPROM se borran mediante rayos ultravioletas de longitud de onda inferior a los 5000 Å. siendo la velocidad de borrado dependiente ligeramente de esta longitud de onda. Puede construirse uno mismo un borrador de EPROMS mediante una lámpara fluorescente de las usadas como germicidas, sin más que ponerle el cebador y la bobina adecuados. Conviene incluirla en una caja cerrada, pues los rayos son altamente dañinos para la vista, pudiendo causar quemaduras y lesiones permanentes.

Memorias EEROM o EAROM (Electrically Erasable ROM o Electrically Alterable ROM)

El borrado del contenido puede hacerse también de forma eléctrica en otro tipo de memorias, simplificándose en gran parte el proceso completo. Sin embargo, las dificultades tecnológicas son mayores, por lo que el número de ciclos de grabación/borrado es todavía pequeño.

Memorias CCD

El almacenamiento transitorio en condensadores se emplea también para otro tipo de memorias, en las que la información es de tipo analógico y de organizado de forma secuencial.

La fuente de información suele suministrarla de forma secuencial, como, por ejemplo, una cámara de vídeo. La información puede almacenarse transitoriamente hasta su transmisión o mantenerse permanentemente, reciclándola desde la salida a la entrada. Veremos una aplicación de este tipo de circuitos en el tratamiento de señales analógicas.

Aplicaciones

Para adquirir un conocimiento práctico de las memorias descritas vamos a presentar algunas aplicaciones que pueden ser de utilidad para cualquier poseedor de ordenador personal.

Ampliación de memoria con RAM estática

La ampliación mediante RAM estática es muy simple para cualquiera de los ordenadores personales utilizados. El mayor problema suele ser la asignación apropiada de la dirección a ocupar. En las máquinas con capacidad de direccionamiento grande siempre suele haber alguna zona libre, pero en las máquinas pequeñas es frecuente que estén los 64 K ocupados y hay que recurrir a algún truco para poder direccionar las nuevas memorias.

Una forma simple de conectarla es como dispositivo periférico, direccionado a través de un bus de direcciones y datos simulado mediante, por ejemplo, la tarjeta de am-

EL TALLER DEL HARDWARE

pliación de puertos. Mediante los registros de salida, cargados con la dirección de la posición a la que queremos referirnos, presentamos al exterior un bus de direcciones simulado. Mediante un puerto de salida en paralelo con uno de entrada simulamos un bus de datos. Las señales de control necesarias las simularemos igualmente con un puerto de salida. La solución puede parecer un poco lenta, pero puede utilizarse sin restricciones en numerosas aplicaciones en las cuales lo que se necesita es disponer de una cantidad adicional de memoria con datos que interesa que no se borren aunque se apague el ordenador. En este caso debería ser memoria EPROM o RAM alimentada con baterías.



Fig. 10. RAM externa conectada a través de puertos.

En el ejemplo vemos un montaje apropiado para una RAM estática, alimentada con pilas recargables, en el caso de apagarse la alimentación general.

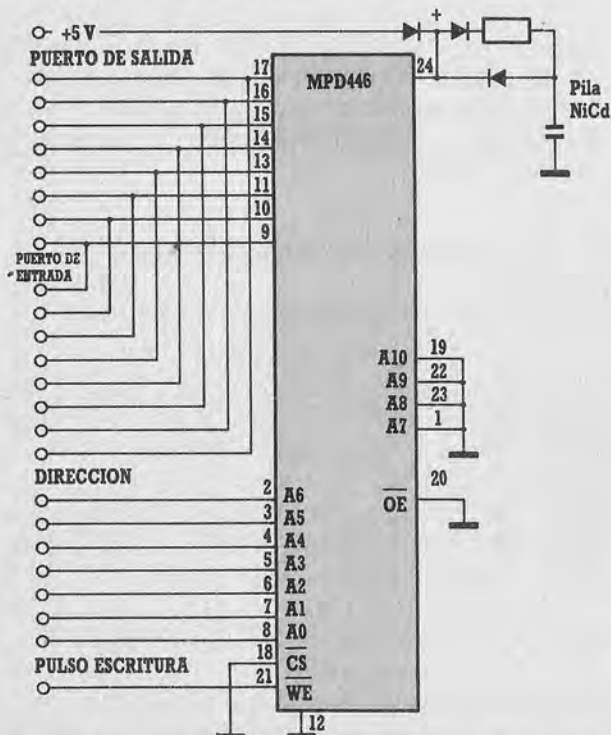


Fig. 11. Circuito eléctrico de RAM externa a través de puertos.

La lectura y escritura de la memoria a través de puerto se realizará mediante la simulación del bus, con instrucciones de OUT para cargar las direcciones y de INP para leer el contenido del bus de entrada.

La forma idónea es conectar la memoria como parte de la zona de direccionamiento propio de la máquina. Si las direcciones coinciden con las de otra memoria RAM ya existente, podemos hacer varias cosas. Una de ellas es seleccionar mediante conmutadores cuál de ellas es la que debe estar activa en cada momento. Otra solución es que los pulsos de escritura y lectura se puedan controlar a voluntad para poder seleccionar cuál de ellas es la que transmite y cuál la que recibe en una operación de transferencia.

Ampliación de memoria con RAM dinámica

Es algo más complicado, si es que no está previsto en el diseño original, pues los pulsos necesarios para el acceso son más complejos. Hay, sin embargo, un caso interesante en el cual es fácil la ampliación de memoria sin gran esfuerzo. Con montaje de unos chips sobre otros, compartiendo todos los pines excepto el de selección, se puede aumentar en 16 ó 64 K la memoria sin necesidad de tarjeta de ampliación ni otra circuitería que un inversor. La modificación del circuito es simplemente la generación de una nueva señal de RAS para los nuevos chips, que debe ser generada para una decodificación correspondiente a una dirección que difiera en el bit más significativo. Así para pasar un equipo con 48 K a 64 K añadiendo una fila de chips 4116, deberemos cablear la patilla 4 de todos los chips en paralelo y conectados a la salida del decodificador de la página más alta de la memoria. Si no se dispone de esta salida será necesario generarla mediante un decodificador, que podría usar la decodificación de los bits A14-A9 ya generado para la página de 48 K.

Ampliación de memoria con EPROM

La ampliación es muy parecida a la de RAM estática, con la diferencia de que no hay que utilizar la señal de escritura. En los ordenadores Spectrum y Amstrad existe una señal en el conector de expansión que permite anu-

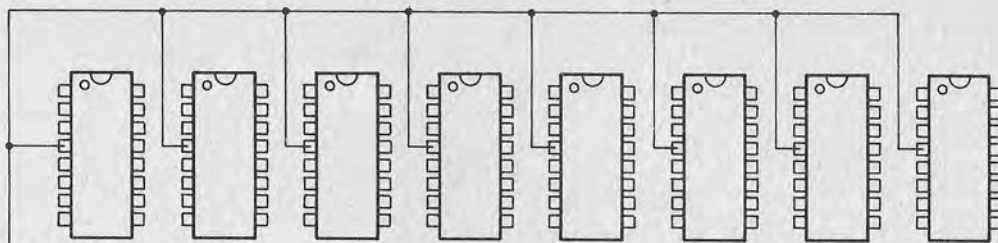


Fig. 12. Ampliación de RAM con montaje sobre chips.

lar la decodificación de la ROM interna. Aunque es delicado, podemos hacer que se ejecute un programa de control propio generado por nosotros, si reproducimos el código del sistema original y le añadimos o modificamos lo que sea necesario. En Commodore, MSX y el IBM-PC pueden insertarse en posiciones específicas ROMS de usuario.

Para los ordenadores en que la nueva memoria sustituye a la original es necesario conectar: todo el bus de direcciones, el bus de datos, la señal de lectura RD y si la memoria es lenta la señal de WAIT para prolongar los ciclos necesarios para la lectura. En el caso del Spectrum la ROM está situada en los 16 K más bajos, es decir, cuando los bits de dirección A15 y A14 están a 0. Se decodifican estas direcciones con la puerta OR y se condiciona con las señales -MREQ y -RD. La ROM interna se bloquea si se introduce una señal de 5 voltios por la entrada ROMCS, por lo que haremos el montaje con un conmutador, para seleccionar la ROM interna o la que se añada. Es necesario tener en cuenta el nivel de activación de la EPROM externa, pues dependiendo del modelo podría necesitar el nivel complementario al utilizado por la del sistema. Se añade un chip de inversores o se utiliza alguno libre de la tarjeta.

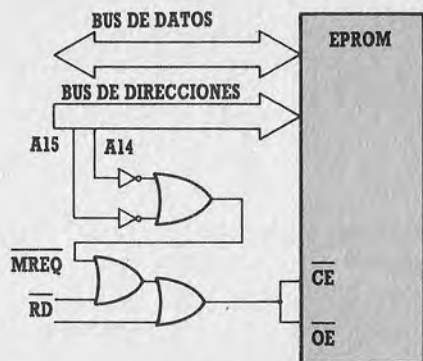


Fig. 13. Ejemplo de sustitución de la ROM por EPROM en Spectrum.

Para poder seguir utilizando la ROM del sistema y poder almacenar rutinas de interés en EPROM, podemos hacer que manualmente sea seleccionable una zona de la memoria para que se comporte de sólo escritura y la EPROM añadida como de sólo lectura. El circuito de lectura de la EPROM es igual al anterior, excepto la ubicación dentro del mapa de memoria. Al conservar la ROM interna hay que situarlo en la parte alta, con lo que entra en conflicto con la RAM. Es necesario bloquear la lectura de la RAM en las posiciones altas, cuando esté en operación la EPROM.

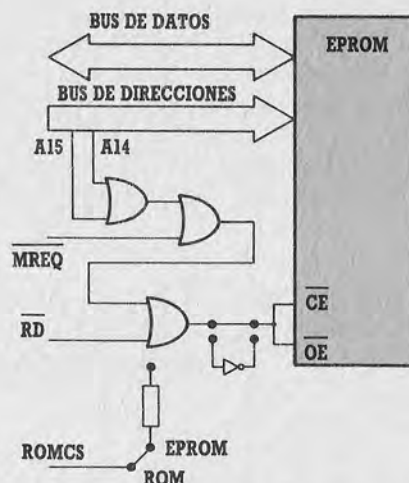


Fig. 14. Memoria EPROM en direcciones altas.

■ Utilización práctica de una memoria EPROM

Podemos emplear una EPROM para múltiples aplicaciones. Como regla general será útil para conversión de código, es decir, generará un código de salida según convenga para uno de entrada dado. Podemos hacer conversiones no lineales y todo a la velocidad de lectura de la EPROM, que siempre será mucho mayor que la posible con las entradas/salidas del ordenador.

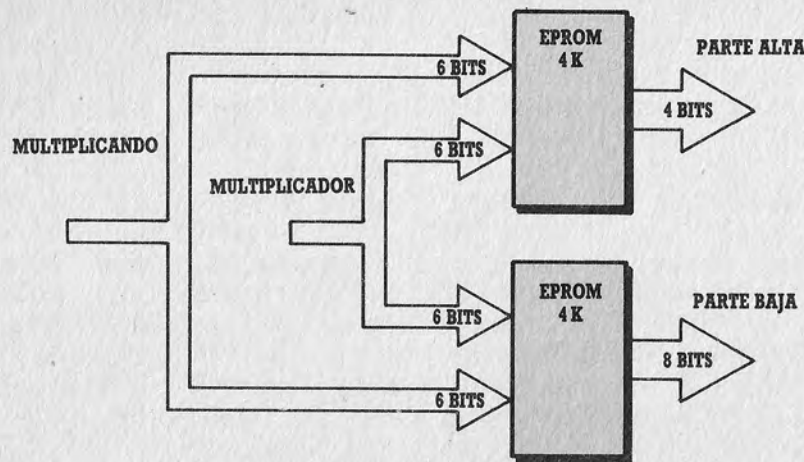


Fig. 15. Aplicación de EPROM a operación aritmética.

Puede incluso utilizarse para realizar operaciones matemáticas, almacenando el resultado para cada dirección indicada por los operandos. En el ejemplo se indica un caso en el que se realiza la operación multiplicación de dos números de seis bits, mediante dos EPROMs de 4 KB. En aplicaciones de proceso de señales puede ser de interés, pues sólo requiere un ciclo, utilizando un componente muy económico. La alternativa con un multiplicador sería mucho más costosa. Tiene como ventaja que puede realizarse la operación con control sobre los límites de los operandos de manera inmediata.

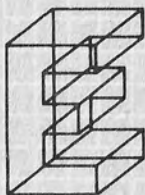
Conclusión

Los diferentes tipos de memorias tienen variadas aplicaciones en nuestros ordenadores y en los dispositivos que se le pueden añadir. Las más interesantes son las memorias EPROM, ya que podemos grabarlas con programas o datos de interés y utilizarlas numerosas veces en nuestras aplicaciones específicas. Para poder grabarlas es necesario contar con un equipo grabador, que puede realizarse con nuestro ordenador personal y unos pocos circuitos adicionales, como veremos en el próximo número.

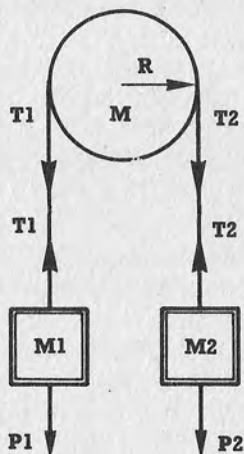
APRENDER CON EL ORDENADOR

NATURALEZA Y TECNOLOGIA

FISICA: Los problemas de poleas



n este apartado vamos a ofrecer un programa que solucione los problemas de física sobre poleas. Supongamos una polea de la que cuelgan dos cuerpos como los de la figura:



Nuestro problema será el de calcular la aceleración con la que bajará el cuerpo de mayor peso (el de masa $M1$) y la aceleración angular de la polea.

Para ello aplicamos las ecuaciones de fuerzas:

- Cuerpo 1: $P1 - T1 = M1 \times a$
- Cuerpo 2: $T2 - P2 = M2 \times a$
- Polea: $T1 \times R - T2 \times R = I \times a/R$

Donde I es el momento de inercia de la polea $= 1/2 \times M \times R \times R$

La Ec. 3 nos queda de esta forma: $T1 - T2 = 1/2 \times M \times a$

Sumando las Ec. 1 y 2 obtenemos: $P1 - T1 + T2 - P2 = M1 \times a + M2 \times a$

y sustituyendo con 3 nos queda: $P1 - P2 = M1 \times a + M2 \times a + 1/2 \times M \times a$; de donde obtenemos que la aceleración será:

$$a = (P1 - P2) / (M1 + M2 + 1/2 \times M)$$

La aceleración angular será esta aceleración dividida entre el radio de la polea.

$P1$ y $P2$ son los pesos de los cuerpos.

$M1, M2$ y M son las masas de los cuerpos y la polea, respectivamente.

R es el radio de la polea y $T1$ y $T2$ las tensiones de la cuerda.

```
10 REM *****
20 REM **
30 REM **      PROGRAMA PARA RESOLVER PROBLEMAS DE FISICA      **
40 REM **      DE POLEAS                                          **
50 REM **
60 REM **      PROGRAMA VALIDO PARA                                **
70 REM **      IBM,MSX,AMSTRAD,SPECTRUM                          **
80 REM **
90 REM *****
```

```

100 SCREEN 1
110 CIRCLE (50,50),20
120 FOR I=50 TO 100
130 PSET (30,I),2
140 PSET (70,I),2
150 NEXT I
160 FOR I=25 TO 35
170 PSET(I,100),2
180 PSET(I,110),2
190 NEXT I
200 FOR I=65 TO 75
210 PSET(I,100),2
220 PSET(I,110),2
230 NEXT I
240 FOR I=100 TO 110
250 PSET(25,I),2
260 PSET(35,I),2
270 PSET(65,I),2
280 PSET(75,I),2
290 NEXT I
300 LOCATE 17,3:PRINT 1
310 LOCATE 17,8:PRINT 2
320 LOCATE 2,15:PRINT "MASA DEL CUERPO 1(EN KG)=":LOCATE 3,15:INPUT M1
330 LOCATE 4,15:PRINT "MASA DEL CUERPO 2(EN KG)=":LOCATE 5,15:INPUT M2
340 LOCATE 6,15:PRINT "MASA DE LA POLEA(EN KG)=":LOCATE 7,15:INPUT M
350 LOCATE 8,15:PRINT "RADIO DE LA POLEA (EN M)=":LOCATE 9,15:INPUT R
360 IF M1>M2 THEN LET C=1:GOTO 390
370 LET AUX=M1: LET M1=M2: LET M2=AUX
380 LET C=2
390 LOCATE 19,1
400 PRINT "EL CUERPO";C;"CAERA CON UNA ACELERACION DE"
410 LET A=(M1*9.8 - M2*9.8)/(M1+M2+M/2)
420 PRINT A;"M/SG"
430 PRINT "LA ACELERACION ANGULAR SERA DE"
440 PRINT A/R;"RAD/SEG"

```

Comentario del programa

El programa dibuja, en primer lugar, un esquema de la polea (sentencias 100-310) y a continuación solicita los datos (sentencias 320-350) calculando la fórmula expresada anteriormente y escribiendo los resultados (sentencias 390-440). En las sentencias 360-380 selecciona el cuerpo de mayor masa, que será el que caerá.

NOTAS IMPORTANTES

— Para el **SPECTRUM**:

- Hay que sustituir todas las instrucciones de la forma PSET(n,m),2 por PLOT n,256-m, siendo n y m los distintos números y variables que aparecen en cada instrucción PSET. El orden de los números o expresiones dentro del PLOT se puede invertir si esta instrucción requiriese, en primer lugar, el número de fila y luego el de columna.

- Sustituir LOCATE por PRINT AT, poniendo en las instrucciones LOCATE de las líneas 320-350 un 10 donde hay un 15 y acortar de alguna forma las frases que aparecen en estas líneas entre comillas. Por ejemplo, poner M1 en vez de MASA DEL CUERPO 1.

- No poner SCREEN 1.
- Poner en 110: CIRCLE 206, 50, 20.

— Para el **AMSTRAD**:

- Sustituir las instrucciones PSET de igual forma que para el SPECTRUM
- No poner SCREEN 1.
- Cambiar el orden de los números de las instrucciones LOCATE.

- Sustituir la instrucción 110 por:

```

110 FOR Y=0 TO 360
112 X=(3.14*Y)/180
114 PLOT INT(50 +20*COS(X)),INT(206-20*SIN(X))
116 NEXT Y

```

- Si el PLOT requiere primero la fila y luego la columna, se deben intercambiar las expresiones de éste.

— Para el **COMMODORE**:

- Suprimir las instrucciones PSET, LOCATE y CIRCLE, ya que en este ordenador no podemos realizar el dibujo de la polea por no existir dichas instrucciones.

- Se aconseja poner al principio del programa una instrucción de borrado de pantalla ..

MATEMATICAS

■ Tabla de números primos

En este apartado vamos a tratar un tema que resulta necesario a la hora de resolver nu-

merosos problemas matemáticos: el de los números primos.

Por definición, sabemos que un número primo es aquel que únicamente es divisible por sí mismo y por la unidad. En ocasiones se nos plantea conocer si un número es primo o no o cuántos números primos hay entre dos

números dados. El programa que se presenta a continuación nos resuelve esto, ya que mediante él podemos obtener todos los números primos que haya entre dos números. Cuando queramos saber únicamente si un número es primo o no bastará con introducir como límites los números anterior y posterior a él.

```
10 REM *****
20 REM **      PROGRAMA PARA HALLAR LA TABLA DE NUMEROS PRIMOS      **
30 REM **      ENTRE DOS NUMEROS DADOS                             **
40 REM **      VALIDO PARA AMSTRAD,COMMODORE,SPECTRUM,MSX,IBM        **
50 REM **      PROGRAMA REALIZADO PARA                               **
60 REM **      TALLER DE INFORMATICA                                 **
70 REM *****
80 REM LA INSTRUCCION 90 PARA COMMODORE ES :90 PRINT CHR$(147)
90 CLS
100 INPUT "INTRODUCE LOS NUMEROS ENTRE LOS QUE QUIERES CALCULAR LA TABLA":A,B
110 IF A<3 THEN PRINT 1;2;:LET A=3
120 IF (A/2)=INT(A/2) THEN LET A=A+1
130 FOR K=A TO B STEP 2
140 FOR C=3 TO INT(SQR(K))
150 IF K/C = INT(K/C) THEN GOTO 180
160 NEXT C
170 PRINT K;
180 NEXT K
```

Comentario al programa

El programa, que como puede observarse, resulta bastante sencillo, realiza los siguientes pasos:

- Solicita los números entre los que se quiere calcular la tabla de números primos (sentencia 100).

- Si el número inferior es menor que 3, se escriben como números primos el 1 y el 2 y se pone como cota inferior para comenzar el cálculo el 3 (sentencia 110). Esto se hace porque en el cálculo posterior vamos a buscar únicamente como posibles números primos los impares y, por tanto, si queremos que nos escriba también el 2 como número primo tendremos que escribirlo antes.

- A continuación viene lo que es el cálculo de la tabla propiamente dicha: la primera sentencia FOR (130) recorre todos los números impares incluidos entre los límites para ver si son primos. De esto último se encarga precisamente la segunda sentencia FOR (140), que recorre todos los posibles divisores del número que estamos investigando y en 150 comprueba si la división es entera o no. Si lo es, el número en cuestión no es primo y se pasa a probar con el siguiente (next k). En caso de que con ninguno de los posibles divisores se cumpla la condición, terminará el segundo FOR y se escribirá el número en 170, ya que éste es primo.

Nótese que a la hora de buscar los divisores de cada número únicamente buscamos hasta la raíz cuadrada del número (parte entera), ya que si un número no tiene ningún divisor menor que su raíz cuadrada, no tendrá ya ningún divisor. Un número divisible significa que se puede descomponer en el producto de otros dos números y de éstos, necesariamente, uno deberá ser menor que la raíz cuadrada y otro mayor (o los dos iguales a la raíz cuadrada).

SOCIEDAD

■ El problema del camino más corto

Hoy vamos a ver en esta sección cómo resolver con nuestro ordenador un viejo problema que el hombre se ha planteado desde siempre: conocer el camino más corto entre dos puntos incluidos dentro de una red de puntos (ciudades) unidas mediante distintos caminos (carreteras).

El programa que se presenta a continuación solicitará los datos de la red (número de ciudades, nombre de éstas, distancias de las carreteras que las unen) y luego nos dirá cuál es el camino más corto entre dos ciudades cualesquiera que le solicitemos.

APRENDER CON EL ORDENADOR

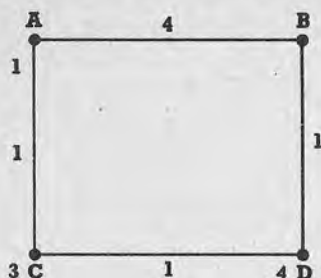


Fig. 2.

Para ver un ejemplo sencillo, supongamos que hemos introducido los datos de la red representada en la figura y a continuación queremos saber el camino más corto para ir de la ciudad A a la ciudad B. El programa nos contestará: A - C - D - B, ya que para ir de A a B es más corto hacer ese recorrido, cuya distancia total es 3, que ir por la carretera que las une directamente, cuya distancia total es 4.

Para la resolución del problema el programa utiliza el algoritmo de Floyd, cuya idea fundamental es la siguiente:

- Se crean 2 matrices cuadradas, C y D, de dimensión $N \times N$ (siendo N el número de ciudades cuyo significado es el siguiente):

$C(i,j)$ = Mínima distancia para ir de i a j.

$D(i,j)$ = Penúltima ciudad en el camino más corto desde i hasta j.

Inicialmente estas matrices valen lo siguiente:

- En $C(i,j)$ se guarda la distancia de i a j cuando existe un camino directo entre i y j y se guarda un valor máximo cuando no existe camino directo.

- En $D(i,j)$ se guarda i, ya que se considera que inicialmente el único modo de ir de i a j es por la carretera que las une y, por tanto, la penúltima ciudad será i.

A continuación el algoritmo estudia si para cada pareja de ciudades de la red existe otra ciudad intermedia unida a ambas, tal que yendo a través de ella el camino entre ambas resulte más corto que el que tenemos como camino más corto (al principio si entre ambas no tenemos un camino directo habremos puesto como distancia más corta el valor máximo y entonces se cumplirá la condición de poder ir a través de la ciudad intermedia). Cuando ocurra esto colocará en el lugar correspondiente de la matriz C la nueva distancia y como penúltimo vértice del camino el penúltimo del camino entre la ciudad intermedia y la final. Después de repetir esto para cada pare-

ja de vértices, al final tendremos en la matriz C las distancias más cortas y en D el penúltimo vértice del camino, y si vamos sacando el penúltimo del penúltimo, sucesivamente obtendremos el camino entero.

Veamos cómo funcionaría el algoritmo para el sencillísimo ejemplo de la figura:



Fig. 3.

Inicialmente la matriz C valdría:

0	1	max.
1	0	2
max.	2	0

y la matriz D:

1	1	1
2	2	2
3	3	3

Al recorrer el algoritmo todas las posibles parejas de ciudades encontrará que entre las ciudades 1 y 3 existe una ciudad intermedia (la 2), tal que la distancia para ir de 1 a 3 es menor pasando por 2 ($1 + 2 = 3$) que la que tenemos puesta hasta ahora (que era la máxima por no existir camino directo entre ellas). Así, cambiaría los valores de las matrices de la siguiente manera:

— Matriz C:

0	1	3
1	0	2
3	0	0

— Matriz D:

1	1	2
2	2	2
1	3	3

Como en nuestro ejemplo ya no hay más parejas de ciudades que cumplan la condición, el algoritmo habría terminado y con estas matrices podríamos ya obtener los caminos más cortos entre cualquier pareja de ciudades.

A continuación se presenta el programa que resuelve este problema.

El programa admite cualquier número de ciudades y cualquier número de carreteras, entre ellas:

```

10 REM *****
20 REM **
30 REM ** PROGRAMA PARA HALLAR EL CAMINO MAS CORTO ENTRE DOS **
40 REM ** CIUDADES INCLUIDAS DENTRO DE UNA RED DE CARRETERAS **
50 REM **
60 REM ** PROGRAMA VALIDO PARA AMSTRAD,IBM,MSX,SPECTRUM,COMMODORE **
70 REM **
80 REM *****
90 CLS
100 INPUT "INTRODUCE EL NUMERO DE CIUDADES QUE FORMAN LA RED";N
110 DIM C(N,N):DIM D(N,N):DIM N$(N)
120 LET MAX=1E+10
130 FOR I=1 TO N
140 PRINT
150 PRINT "TECLEA EL NOMBRE DE LA CIUDAD Nº";I
160 INPUT N$(I)
170 FOR J=1 TO N:LET C(I,J)=MAX:NEXT J
180 LET C(I,I)=0
190 PRINT
200 NEXT I
210 CLS
220 PRINT "INTRODUCE A CONTINUACION LA DISTANCIA ENTRE LAS CIUDADES QUE SE";
230 PRINT "INDICAN CUANDO ENTRE ELLAS HAYA UN CAMINO DIRECTO.";
240 PRINT "SI NO HAY CAMINO DIRECTO PULSA SIMPLEMENTE <RETURN>";
250 PRINT:PRINT
260 FOR I=1 TO N
270 FOR J=I+1 TO N
280 PRINT N$(I);"-";N$(J);
290 INPUT DIS
300 IF DIS=0 THEN GOTO 320
310 LET C(I,J)=DIS:LET C(J,I)=DIS
320 NEXT J:NEXT I
325 PRINT:PRINT "!UN MOMENTO!,ESTOY CALCULANDO"
330 FOR I=1 TO N
340 FOR J=1 TO N
350 LET D(I,J)=I
360 NEXT J:NEXT I
370 FOR I=1 TO N
380 FOR K=1 TO N
390 IF I=K OR C(I,K)=MAX THEN GOTO 440
400 FOR J=1 TO N
410 IF J=K OR C(K,J)=MAX THEN GOTO 430
420 IF C(I,K)+C(K,J)<C(I,J) THEN LET C(I,J)=C(I,K)+C(K,J):LET D(I,J)=D(K,J)
430 NEXT J
440 NEXT K:NEXT I
450 GOSUB 1000
460 PRINT:PRINT
470 INPUT "INTRODUCE LOS NUMEROS DE DOS CIUDADES (SEPARADOS POR COMAS) ENTRE
LAS QUE QUIERAS SABER EL CAMINO MAS CORTO.";C1,C2
480 PRINT:PRINT N$(C1);" - ";N$(C2):PRINT
490 LET DIS=C(C1,C2)
500 PRINT N$(C1);" - ";
510 IF C1=C2 THEN PRINT:PRINT:GOTO 535
520 LET C1=D(C2,C1)
530 GOTO 500
535 PRINT "la distancia minima es";DIS
540 INPUT "QUIERES TERMINAR(S/N)";R$
550 IF R$="N" THEN GOTO 450
560 IF R$="S" THEN END
570 GOTO 540
1000 CLS
1010 FOR I=1 TO INT(N/2)
1020 PRINT (I*2)-1;N$((I*2)-1);TAB(40);I*2;N$(I*2)
1030 NEXT I
1040 IF N/2<>INT(N/2) THEN PRINT N;N$(N)
1050 RETURN
1060 REM *****
1070 REM ** MODIFICACIONES PARA SPECTRUM: **
1080 REM ** 110 DIM C(N,N):DIM D(N,N):DIM N$(N,10) **
1090 REM ** 560 Cambiar END por GOTO 9999 **
1100 REM ** MODIFICACIONES PARA COMMODORE: **
1110 REM ** Cambiar CLS POR PRINT CHR$(147) en 90,210 y 1000 **
1120 REM *****

```

Comentario del programa

El programa comienza pidiendo el número de ciudades y los nombres de éstas (que

guarda en N\$). Aprovechando la sentencia FOR de la línea 130 que recorre desde 1 hasta el número de ciudades para pedir los nombres de éstas, se dan también en ella los va-

APRENDER CON EL ORDENADOR

lores iniciales de la matriz C (líneas 170 y 180).

A continuación el programa sacará por pantalla todas las parejas posibles de ciudades para que le introduzcamos las distancias entre las carreteras que las unen. Cuando no haya carretera entre ellas (que las una directamente), se pulsará simplemente RETURN.

El siguiente paso (sentencias 370-440) es la realización por parte del programa del algoritmo explicado anteriormente y luego ya nos pide parejas de ciudades para obtener el camino más corto entre ellas (sentencias 500-530).

PARA LOS MAS JOVENES

■ Aprender nombres de plantas, flores y árboles jugando

Hoy presentamos en esta sección un juego para que los más pequeños se vayan familiarizando con distintos nombres de plantas, flores y árboles a través de un juego muy popular consistente en adivinar una palabra conociendo la letra por la que empieza, la letra por la que termina y una definición. El juga-

```
10 REM *****
20 REM **
30 REM ** PROGRAMA PARA ACERTAR NOMBRES DE PLANTAS A **
40 REM ** PARTIR DE UNA DEFINICION UTILIZANDO EL JUEGO **
50 REM ** CONSISTENTE EN IR ADIVINANDO LAS LETRAS **
60 REM **
70 REM ** VALIDO PARA AMSTRAD,IBM,MSX,COMMODORE **
75 REM ** ESTE PROGRAMA NO ES VALIDO PARA SPECTRUM **
80 REM *****
90 DIM P$(20,11):DIM L$(11)
95 REM LA INSTRUCCION 100 SOLO ES NECESARIO UTILIZARLA PARA MS-DOS (IBM 0
    COMPATIBLES)
100 RANDOMIZE TIMER
110 DATA 84,82,73,71,79,0,67,69,66,65,68,65,0,65,86,69,78,65,0,71,65,82,66,65,78
120 DATA 90,79,83,0,74,85,68,73,65,83,0,71,85,73,83,65,78,84,69,83,0,65,76,67,65
130 DATA 67,72,79,70,65,0,84,79,77,65,84,69,0,90,65,78,65,72,79,82,73,65,0,78,65
140 DATA 82,65,78,74,65,0,76,73,77,79,78,0,80,79,77,69,76,79,0,77,69,76,79,67,79
150 DATA 84,79,78,0,67,69,82,69,90,65,0,65,66,69,84,79,0,82,79,66,76,69,0,65,66
160 DATA 69,68,85,76,0,80,69,78,83,65,77,73,69,78,84,79,0,79,82,81,85,73,68,69
170 DATA 65,0,65,77,65,80,79,76,65,0
180 PRINT :PRINT "¡UN MOMENTO!, ESTOY CARGANDO LOS DATOS"
190 FOR I=1 TO 20
200 LET J=1
210 READ A
220 IF A=0 THEN FOR K=J TO 11 :LET P$(I,K)=" ":NEXT K :GOTO 250
230 LET P$(I,J)=CHR$(A)
240 LET J=J+1:GOTO 210
250 NEXT I
260 LET PAL=INT(RND(1)*20)+1
270 FOR J=1 TO 11 :LET L$(J)=" ":NEXT J
280 LET FT=0 :LET CON=0
290 CLS
300 ON PAL GOTO 310,310,310,320,320,330,340,350,360,360,360,370,380,390,400,
    410,420,420,430
310 PRINT "CEREAL CULTIVADO AMPLIAMENTE EN TODA EUROPA":GOTO 440
320 PRINT "LEGUMBRE CULTIVADA EN LOS PAISES PROXIMOS AL MEDITERRANEO Y A LA
    COSTA ATLANTICA":GOTO 440
330 PRINT "HORTALIZA ":GOTO 440
340 PRINT "HORTALIZA EMPLEADA EN LA CONFECCION DE ENSALADAS":GOTO 440
350 PRINT "HORTALIZA QUE CRECE BAJO TIERRA":GOTO 440
360 PRINT "FRUTA TIPICA DE ZONAS MEDITERRANEAS DE LA QUE SE OBTIENEN EXCELENTE
    ZUMOS":GOTO 440
370 PRINT "FRUTA ":GOTO 440
380 PRINT "FRUTA DE COLOR ROJIZO EMPLEADA EN REPOSTERIA":GOTO 440
390 PRINT "ARBOL DE HOJA PERENNE UTILIZADO COMO ARBOL TIPICO NAVIDEÑO":GOTO 440
400 PRINT "ARBOL DE HOJA CADUCA MUY EXTENDIDO EN TODOS LOS BOSQUES EUROPEOS":
    GOTO 440
410 PRINT "ARBOL CUYA SAVIA ES UTILIZADA EN ALGUNOS PAISES PARA PREPARAR
    BEBIDAS":GOTO 440
420 PRINT "PLANTA QUE DESTACA POR LA BELLEZA DE SUS FLORES":GOTO 440
430 PRINT "FLOR CONOCIDA POR SU BRILLANTE COLOR ROJO"
440 LET L$(1)=P$(PAL,1)
450 FOR I=11 TO 2 STEP-1
460 IF P$(PAL,I)<>" " THEN LET L$(I)=P$(PAL,I):LET F=I:GOTO 480
```

```

470 NEXT I
480 LOCATE 10,10
490 FOR I=1 TO F
500 PRINT L$(I); " ";
510 NEXT I
520 PRINT:PRINT TAB(12);
530 FOR I=2 TO F-1
540 PRINT "- ";
550 NEXT I
555 REM PARA AMSTRAD :560 LOCATE 1,15
560 LOCATE 15,1
570 IF FT THEN GOTO 690
580 LET FT=1
590 PRINT "TECLEA LAS LETRAS QUE CREAS QUE FALTEN EN LA PALABRA(MAYUSCULAS)"
600 LET A$=INKEY$
610 IF A$="" THEN GOTO 600
620 FOR I=2 TO F-1
630 IF P$(PAL,I)=A$ THEN LET L$(I)=P$(PAL,I):LET LT=1
640 IF L$(I)=" " THEN LET FT=0
650 NEXT I
660 IF LT=0 THEN CON=CON+1 :PRINT:PRINT"LO SIENTO,LA LETRA ";A$;" NO ESTA"
670 LET LT=0
680 GOTO 480
690 PRINT "
700 PRINT :PRINT "
710 PRINT ";ENHORABUENA!,LA HAS ACERTADO":PRINT
720 PRINT "HAS TECLEADO ";CON;" LETRAS ERRONEAS AL INTENTAR ACERTAR LA PALABRA"
730 PRINT "QUIERES JUGAR OTRA VEZ? (S/N)"
740 LET A$=INKEY$
750 IF A$="S" THEN GOTO 260
760 IF A$(">N" THEN GOTO 740
780 REM *****
790 REM ** MODIFICACIONES PARA COMMODORE : **
800 REM ** 290 PRINT CHR$(147) **
810 REM ** 480 FOR I=1 TO 8 **
820 REM ** 482 PRINT **
830 REM ** 484 NEXT I :PRINT TAB(10); **
840 REM ** 560 FOR I=1 TO 4 **
850 REM ** 563 PRINT **
860 REM ** 565 NEXT I **
870 REM ** 600 GET A$ **
880 REM ** 680 GOTO 290 **
890 REM ** 740 GET A$ **
900 REM *****

```

dor a partir de esto deberá ir introduciendo las letras que crea que faltan en la palabra una a una hasta que la complete. Cuando esto haya ocurrido el programa dirá los fallos que se han cometido al intentar acertarla.

Comentario del programa

El programa comienza con 7 instrucciones DATA en las que se introducen los datos correspondientes a las palabras que posteriormente habrá que adivinar. Como se observará a la vista del programa, los datos de estas instrucciones no se parecen lo más mínimo a una lista de palabras, sino que son una lista ordenada de números. Estos números precisamente se corresponden con los códigos internos que las letras tienen dentro de nuestro ordenador. Cada número es una letra. La razón de introducir en los DATA los códigos de las letras en vez de las propias letras directamente es obvia: si introdujésemos ahí las palabras, al teclear el programa en nuestro ordenador

tendríamos que teclear las palabras que posteriormente al ejecutar el programa tendríamos que adivinar, con lo que el juego perdería su interés.

El programa guarda las palabras que lee, después de transformar los códigos en letras mediante la función CHR\$ en la tabla p\$. El número de palabras de que consta el juego es 20. El programa selecciona mediante la función RND (sentencia 260) un número al azar entre 1 y 20, que será el número de la palabra que hay que adivinar. Entonces escribe la definición mediante la sentencia ON (línea 200) para posteriormente pedirte las letras que faltan.

El programa guarda en la variable L\$ la palabra con las letras que tú ya has adivinado, teniendo blancos en las letras que aún te quedan por acertar. Así, entre las sentencias 440 y 470 guarda en L\$ la primera y última letra de la palabra en sus lugares correspondientes. Cuando introduces una letra, comprueba si está, y si es así la guarda en su lu-

APRENDER CON EL ORDENADOR

gar correspondiente de L\$ (líneas de 620 a 650) para luego volver a 480, donde escribe de nuevo L\$ con las letras introducidas.

Significado de algunas variables de control que utiliza el programa:

LT = 0 indica que la última letra introducida no está en la palabra

= 1 lo contrario

FT = 1 Indica que se ha acertado la palabra

=0 La palabra todavía no se ha acertado

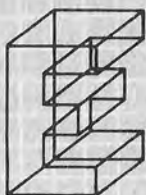
F Número de letras de la palabra en cuestión

Las variables LT y FT se ponen a 0 y 1, respectivamente, cada vez que empezamos una nueva operación y si luego se comprueba que lo que indican estos valores no se cumple, se les cambia de valor.

Este programa no es válido para el SPECTRUM debido a que no posee la instrucción ON GOTO que se utiliza en él. Se deja como ejercicio a algún avisado lector su modificación para que haga lo mismo sin esta instrucción para que pueda ser ejecutado en el SPECTRUM.

PEQUEÑA HISTORIA DE LA INFORMATICA

■ Historia del valle del Silicio. Silicon valley



El valle de Santa Clara, o valle del Silicio, se extiende en California, protegido por las montañas de Santa Cruz, en la península de San Francisco. Abrigado por el sistema montañoso, el valle tiene un clima suave, tipo mediterráneo,

con una vegetación tropical o subtropical (robles, olivos, vides, etc.). Podría pensarse en su suelo como pobre, y seco, pero los vientos de la zona producen un microclima lluvioso durante dos o tres meses al año, lo que resulta suficiente para que el valle sea verde y fértil. Es un pequeño paraíso tranquilo.

Sus primeros pobladores eran unos indios muy pacíficos, los ahones, probablemente llegados de Asia atravesando desde las islas Aleutianas, hace más de cinco mil años. (Entonces existía un brazo de tierra firme que permitía esas migraciones.) Estos indios eran muy tranquilos, y apenas si se desarrollaron en los siglos que vivieron en esos valles. Tenían mucha agua, caza, un clima benigno, por lo que no necesitaron poner a prueba su imaginación para mejorar su vida. Este escaso progreso puede comprobarse en los vestigios que han quedado de ellos, en sus tumbas y herramientas. (El proceso es el mismo que el de los animales que pueblan Oceanía; ellos tampoco se han desarrollado como los del res-

to del mundo, debido a las condiciones fáciles de su vida, que los ha hecho débiles frente a los demás.)

Los españoles llegaron a aquellas tierras, principalmente con intenciones catequistas. Rebautizaron el valle, naturalmente, con nombres de santos: Santa Clara (precioso nombre en recuerdo de Clara, hija de los condes de Sasso Rosso y seguidora de San Francisco), San José, Santa Cruz, San Francisco. De los indios, al cabo de varias generaciones quedó poco, ya que no eran fuertes, ni podían acostumbrarse a las nuevas formas de vida. Además, fueron muy sensibles a las nuevas enfermedades traídas por los españoles. Sin embargo, fueron las leyes y costumbres de los pobladores americanos las que acabaron, finalmente, con los indios.

Más tarde los mexicanos, tras su independencia de España, perdieron grandes extensiones en la guerra contra los Estados Unidos, y como veremos también se vieron forzados a vender enormes territorios.

Los primeros nuevos colonos de la zona eran pequeños terratenientes, que cultivaban huertas de poca extensión, que no les daban excesivo trabajo (el suelo era muy fértil), con lo que, en general, no eran muy ambiciosos.

Con la afluencia enorme de emigrantes a la costa este de los Estados Unidos, éstos comenzaron a aplicar técnicas de repoblación de las enormes zonas sin habitar.

Construyeron el ferrocarril hacia el oeste, y algunas carreteras. Con esta política co-

PEQUEÑA HISTORIA DE LA INFORMATICA

menzaron a llegar muchos pobladores a la zona, especialmente durante la década de 1840. En 1846 el Gobierno de los Estados Unidos pagó 15 millones de dólares al Estado de México por grandes extensiones en California.

Sin embargo, ocurrió un acontecimiento que cambió totalmente la tranquilidad del valle. Apareció oro. Y con él, multitud de gentes, por lo general sin preparación, dispuestas a enriquecerse de la noche a la mañana. Como el oro se encontró más al sur, cerca de Sacramento, los habitantes del valle dejaron en él a sus mujeres e hijos, dirigiéndose a la zona privilegiada. El valle quedó abandonado, y ya no volvió a resurgir hasta más de un siglo después, con la aparición de los semiconductores.

La Universidad de Stanford, situada en la zona, y de tan gran prestigio, comenzó a funcionar en 1891, en unos terrenos y con donaciones de Leland Stanford, uno de los padres del ferrocarril norteamericano. Sin embargo, nadie predeciría en sus comienzos que podría competir en investigación con las mejores universidades mundiales.

Una vez establecida la "filiación" del valle de Santa Clara, pasemos al valle del Silicio. Aunque se trate del mismo valle físico que el de Santa Clara, desde otros puntos de vista son dos valles totalmente distintos. La zona, desde el punto de vista industrial, no tenía ningún atractivo. Naturalmente, tenía silicio, pero ese elemento existe en nuestro planeta en cantidades ingentes. Su interés surgió simplemente a raíz de que se estableciera en él el inventor del transistor.

Desde finales de los años cincuenta, y siguiendo unas compañías el ejemplo de otras, se han ido estableciendo en el valle gran cantidad de firmas dedicadas a semiconductores, electrónica e informática. La llegada de sus dirigentes, hombres por lo general bien preparados y con un espíritu emprendedor, ha configurado el valle, convirtiéndolo en otra cosa totalmente distinta. La mitad de la población del valle tiene estudios universitarios y, por el contrario, no tienen especiales ataduras familiares, ya que la mayoría procede de otras zonas. Por tanto, sólo cuentan con la familia estricta de mujer e hijos. Nuestro valle ya no es el valle tranquilo de antes, ahora, en las preciosas calles y jardines pacíficos de hace cien años, se yerguen fábricas, bullen posibilidades, inversiones, fracasos, éxitos. En este valle un hombre puede hacerse millonario en cuatro o cinco años, o, por el contrario, puede

Sabía usted que...

El profesor G. H. C. Macgregor, de la Universidad de Glasgow, y el reverendo Morton, de Culross, condado de Fife, estaban muy interesados en descubrir si una determinada epístola era o no de san Pablo. Basándose en que las personas tienen determinados hábitos que se ven reflejados también en su estilo literario, se embarcaron en la tarea de identificar los distintos manierismos de las epístolas. Para el trabajo utilizaron un ordenador Mercury, y procesaron más de 400 textos de prosa griega para detectar las apariciones de determinados modismos.

Después de otro período en el que hubieron de dedicarse a otras tareas, volvieron a su investigación, llegando a conclusiones muy interesantes: cinco de las catorce epístolas (a los romanos, la dos de los corintios, la de los gálatas y la de Filemón) habían sido escritas por la misma mano. Las nueve restantes habían sido redactadas por al menos cinco personas distintas. Si san Pablo es autor reconocido de la epístola a los gálatas, de ello se deduce que lo es también de las cuatro otras epístolas mencionadas, no siendo auténticas las restantes.

fracasar y arruinarse. Sin embargo el fracaso como tal no tiene importancia, es un detalle más de la vida de una persona. Lo importante es cómo ha aceptado el fracaso y cómo ha resurgido de las cenizas.

Todo es efímero en el valle. Los proyectos costosísimos en investigación tienen una validez de dos, tres años. Es una carrera vertiginosa para adelantar a los demás, para innovar, mejorar antes que el vecino de enfrente te adelante y deje los productos obsoletos.

El personal (los ingenieros) que trabajan en una determinada firma, pueden abandonarla si reciben otra oferta mejor de trabajo, o si no forman su propia empresa, como han hecho tantos y tantos pioneros de la Informática. La seguridad en el trabajo prácticamente no existe. Sin embargo, tampoco el fracaso es estrepitoso. La mayoría poseen conocimientos como para emplearse en buenas condiciones en otra empresa del valle.

Las empresas que se han establecido en la zona (naturalmente, las que no han quebrado) han crecido vertiginosamente. Sus edificios, de cristal y plástico, se han ido ampliando con las necesidades. Sin embargo, y a pesar de lo que pueda parecer a primera vista, los empresarios de Silicon Valley no trabajan y arriesgan su dinero para obtener más dine-

ro. Evidentemente, la obtención de dinero es un objetivo primario, pero el más importante es la creación. Esta afirmación se basa en que muchos de ellos alcanzan niveles económicos altísimos, con los que podrían retirarse tranquilamente a disfrutar de la vida. Sin embargo, son raros los que lo hacen. Siguen invirtiendo en nuevos productos. El enorme dinamismo existente no podría existir basado en estructuras jerárquicas anquilosadas y antiguas. En este valle gentes de tanto renombre y reputación como William Hewlett y David Packard piden a sus empleados que les llamen por sus nombres de pila; no existen apenas comedores ni instalaciones reservadas a directivos, ni éstos están aislados siempre en magníficos despachos con secretarías personales. Cada uno hace su trabajo y accede a la persona que necesita (puede llegar, incluso, al propio empresario).

Quizá lo que conforme a estos hombres asentados en el valle es su capacidad de arriesgarlo todo a una carta, por el propio afán de arriesgarse y avanzar en nuevas creaciones. (En el fondo, no cuentan nunca con el fracaso, y, como antes indicamos, tampoco les supone el suicidio de sus carreras.)

Las primeras empresas afincadas en el valle fueron de semiconductores. Unos años después, con la aparición de la integración, muchas empresas fabricantes de chips se animaron a fabricar ordenadores. A raíz de tanta tecnología, fue obvio que había un campo abonado para los creadores de software, y se establecieron numerosas firmas. Hoy en día existen muchas empresas muy especializadas en aplicaciones de los ordenadores para la medicina, y otras ciencias, enormes bancos de datos, etc.

Uno de los primeros personajes que se establecieron en el valle fue Robert Noyce. El Shockley Semiconductor Laboratory, donde trabajaba, fue una de las primeras empresas radicadas en la zona. Se instaló en la ciudad de Palo Alto (su nombre se debe a una enorme secuoia que existía en el pueblo). Antes que esta empresa se habían instalado otras, como Hewlett Packard, pero no puede decirse que el valle tuviera entidad propia hasta que se inventaron los semiconductores.

Shockley, uno de los padres del transistor, obtuvo en 1956 el Premio Nobel de Física, junto con Bardeen y Brattain. Era un hombre brillante, nacido en la zona que estudió en Stanford, donde su padre era profesor. Años más tarde



comenzó a trabajar en los laboratorios de la Bell. En esa época, la Bell estaba interesada en mejorar su red telefónica, ya que los amplificadores no eran en absoluto fiables, se estropeaban muy fácilmente, gastaban mucha energía y eran ruidosos. Shockley comenzó a trabajar con otro tipo de dispositivos, que no utilizaran el vacío (no en vano había hecho su tesis sobre la Física del estado sólido). Experimentó con muchos materiales, y observó que el germanio, silicio entre otros, no eran conductores, pero que al introducirles pequeñas impurezas, pasaban a conducir la electricidad. Este descubrimiento ha sido trascendental, y en él se han basado innumerables técnicas posteriores. Cuando los semiconductores comenzaron a utilizarse masivamente, una vez salvados los primeros obstáculos, Shockley se decidió a fundar su propia empresa, y la situó en Palo Alto, cerca de donde había nacido y estudiado.

El sentido comercial de Shockley no era, sin embargo, tan brillante como su capacidad de investigación. Su empresa no funcionó bien. Además, tenía un carácter celoso e irascible, lo que le hacía antipático a la mayoría de sus colaboradores, que se iban marchando de la empresa. Cada partida de uno de ellos era considerada una "deserción", y el ingeniero correspondiente pasaba a la interminable lista de "enemigos" de Shockley.

Uno de los que decidió abandonar a Shockley fue Robert Noyce (el que más tarde sería el padre del chip). Se marchó, junto con otros siete colaboradores, de Shockley, cansados de los desaires y del aire mortecino que iba tomando la empresa.

Los nuevos empresarios no disponían de dinero para comenzar su singladura, por lo que, tras varios intentos, tuvieron que recurrir

PEQUEÑA HISTORIA DE LA INFORMATICA

a un industrial muy arriesgado, Sherman Fairchild. Fairchild pondría el dinero, y sus asociados dirigirían la fábrica, pero él tendría una opción para la compra de la compañía.

En 1959 Fairchild compró la compañía por tres millones de dólares, y sus coasociados pasaron a ser empleados de la Fairchild. Sin embargo, una vez probada la manzana del empresariado, la mayoría de los antiguos copropietarios decidieron probar suerte de nuevo. De este modo, la Fairchild parecía hacer honor a su nombre, dejando partir a sus mejores hombres. De ella nacieron multitud de empresas: Amelco y Signetics, en 1961; Molectro, en 1962; General Microelectrónica, en 1963; National Semiconductor, en 1967, y muchas otras más (en total, más de cincuenta).

Por fin, en 1968, Noyce abandonó la Fairchild con otros compañeros. Había estado dirigiéndola desde 1961, a raíz de la primera partida de su compañero Baldwind, para fundar otra empresa. Y deseaba volver a la labor de investigación. La empresa fundada se llamó Intel, y se dedicó fundamentalmente al desarrollo de las ideas de integración en las que había trabajado Noyce en la Fairchild. Comenzaron por utilizar chips también como dispositivo de memoria, además de hacerlo para operaciones de cálculo.

Poco tiempo después, una empresa japonesa les pidió una calculadora sencilla que llevara chips. Ted Hoff conectó cuatro chips y los programó para que realizaran las operaciones básicas de cálculo. La idea tuvo éxito, y fue el gran paso del circuito integrado al microprocesador.

Esta primera estructura elemental se llamó MCS4, pero más adelante sufrió otra integración en otro circuito integrado único, el 8008.

Bob Noyce sigue en la Intel, y de su empresa también han salido otras empresas hijas, como, por ejemplo, la Apple. Esta firma fue creada por el antiguo director de marketing de la Intel, Mike Makkula, y otros dos ingenieros, Steve Wozniak y Steve Jobs. Parece que a Noyce no le gustó en absoluto la creación de esta nueva empresa, pero el afán de independencia y creación de los ingenieros del valle crea ese equilibrio inestable, al que, por lo demás, parecen ya acostumbrados sus habitantes.

Hasta ahora no hemos hablado de uno de los verdaderos "creadores" del valle del Silicio: Frederick Termann.

Termann fue quizá el primer hombre que creyó en el potencial humano de la zona. Su vida fue bastante complicada, y de ella hablaremos con más detalle más adelante. Baste decir que fue, entre otras cosas, profesor de la Universidad de Stanford. Entre sus alumnos favoritos estaban William Hewlett y David Packard, que trabajaban en pequeños experimentos de radio en la buhardilla de su casa. Sin embargo, en 1934 a Packard le ofrecieron un puesto en el este, en la General Electric, y partió. Por el momento, en la otra costa había unas esperanzas de futuro mucho mejores.

Hewlett prosiguió sus investigaciones en radio, en el valle, diseñando un oscilador de señales de audio. El invento le abrió los ojos a muchas otras posibilidades comercializables, que estaban a su alcance, pero para ello era necesario obtener dinero, y contar con algún otro colaborador. Convenció a Packard para que volviera del este (excedente de General Electric). Su amigo sufrió un bajón considerable en su sueldo, que pasó a menos de la mitad. Su cómodo despacho se modificó también ligeramente: los dos investigadores instalaron el negocio en el garaje de Packard. Hoy en día es una de las mayores empresas del valle del Silicio. Está ubicada en Palo Alto y factura más de cinco mil millones de dólares anuales. Los dos empresarios todavía recuerdan la película *Fantasia*, cuya banda de sonido se grabó con tecnología desarrollada por la incipiente Hewlett Packard.

Posiblemente, una de las razones básicas de la existencia del valle del Silicio es la Universidad de Stanford. Es verdad que muchos ingenieros que trabajan en el valle se han graduado en el MIT (Massachusetts Institute of Technology), en Harvard y en otras Universidades. Sin embargo, desde que comenzaron a asentarse las primeras empresas de semiconductores, ha existido siempre una conexión enorme entre estudiantes, profesores y empresarios. Esa unión difícil entre los conocimientos teóricos y los prácticos es, probablemente, una de las claves del éxito de este paraíso en continuo movimiento.

En el próximo capítulo seguiremos hablando del mismo valle. De las personas que financiaron las empresas y de otros investigadores y empresarios asentados en la zona. También hablaremos de cómo el presidente Reagan, actor, llegó a gobernador de California, y de empresarios que dirigen su empresa situado a 700 km. de distancia.

■ Los ordenadores de la quinta generación (II)



OMENCEMOS con un pequeño resumen. Como antes vimos, los ordenadores de la quinta generación van a suponer en cierto modo una ruptura con las antiguas concepciones (aunque coexistirán con ordenadores anteriores y otros modernizados y mejorados, contruidos bajo las mismas o parecidas estructuras). Fundamentalmente se caracterizarán por los rasgos siguientes:

- a) Serán capaces de almacenar información en memorias rápidas y fiables de fácil acceso, y enormes si se comparan con las actuales. (Técnicas de integración a gran escala.)
- b) Serán capaces de resolver problemas. Obtendrán los conocimientos necesarios, de grandes bases de datos relacionales, y siguiendo procedimientos inductivos, llegarán a una o varias posibles soluciones de los problemas.
- c) Dispondrán de un interfaz con el usuario, de forma que éste pueda comunicarse con la máquina en lenguaje natural (de viva voz, por medio de imágenes, etc.).

Todo lo anterior constituye un pequeño «prototipo» del ordenador de la quinta generación, de uso general.

Además, se crearán enormes redes, intercomunicadas, que accederán a grandes «almacenes» de saber, para tomar las decisiones

oportunas. Dispondrán de sistemas de localización de averías, etc.

Evidentemente, para el año 1991 no se habrán cubierto todos los objetivos. Se trata de un plan gradual, aunque ambicioso. Algunas de las metas se habrán superado, o al menos esbozado, aunque los perfeccionamientos se vayan realizando paulatinamente en los años posteriores.

Los procesadores, por otra parte, utilizarán tecnología Josephson (que tiene enormes prestaciones y con ella se consigue, sobre todo, aumentar la velocidad, que puede llegar a ser de 10 a 100 veces superior a la alcanzada, por ejemplo, por el IBM 3033). Además, los tiempos de conmutación de esta tecnología son muy buenos, y la disipación de potencia mínima. Todo ello hace a estos procesadores óptimos para ordenadores muy avanzados. Se utilizarán tecnologías MOS (de óxido de silicio) y otras técnicas bipolares.

En el Instituto Tecnológico de California se han desarrollado unas técnicas muy depuradas de utilización de litografías o plantillas para la realización de circuitos de integración a gran escala. Son laminillas multiproyecto, ya que sirven para la creación de muchos microcircuitos.

El salto o ruptura de estas máquinas de la quinta generación se basa fundamentalmente en su arquitectura. La arquitectura de estos nuevos ordenadores debe ser muy flexible para adaptarse a diferentes aplicaciones. La de von Neumann no cumple estas características, y por todo lo expuesto, se utilizarán otras arquitecturas mejor adaptadas al manejo de información relacionada con los «conoci-

TEMAS MONOGRAFICOS DE VANGUARDIA

mientos», con el «saber humano». Las cualidades deductivas de estos ordenadores serán considerables (de cien a mil inferencias lógicas por segundo, siendo necesarias unas cien instrucciones del tipo actual para describir una de estas inferencias lógicas). Por tanto, el resultado es semejante a disponer de cien procesadores a un mismo tiempo. A pesar de todo lo dicho, existirán ordenadores, sobre todo calculadores, que tendrán esta estructura junto a otras características modernísimas y también existirán máquinas dedicadas o especializadas en tareas concretas.

El doctor Fuchi, de Japón, divide la estructura de las máquinas de la quinta generación en siete niveles, los niveles superiores manejan la aplicación y el interfaz hombre-máquina el lenguaje natural. Los restantes manejan las funciones de base y controlan los elementos físicos del sistema.

El procesador de conocimientos dispone de mecanismos de inferencia y sirve de enlace entre la máquina y el interfaz hombre-máquina. Funciona a partir de lenguaje natural.

El procesador es el «núcleo» del ordenador del futuro y se programará en PROLOG, como lenguaje de programación, manejando silogismos de Horn. El PROLOG es el lenguaje que los japoneses han considerado que mejor se adapta a los mecanismos de inferencia lógica. Sin embargo, pasarán muchos años hasta que esté absolutamente desarrollado. El procesador de la base de datos puede tener la arquitectura tradicional de von Neumann. Los restantes procesadores pueden tener arquitecturas paralelo o pipeline (conducto).

El procesador de conocimientos es una parte importante del ordenador. Una de sus cualidades fundamentales será la congruencia. Su labor es ir adquiriendo conocimientos, buscando que éstos no invaliden los anteriormente adquiridos y utilizados. Dispondrá de razonamiento inductivo, es decir, podrá sacar conclusiones a partir de unos datos. Este tipo de problemas están relacionados con algunos de «aprendizaje», de Robótica.

En este tipo de lenguajes se utilizarán técnicas de programación sistemática basadas en abstracción de datos. Además, los procesos de simbolismos y el desarrollo de algoritmos se simplifican notablemente.

Estudiemos, durante unos momentos, un capítulo que aparece siempre en cualquier problema humano: el presupuesto, es decir, el dinero necesario.

Para el trienio 1982-1984, el presupuesto es de 79.000 millones de pesetas. Como antes expusimos, aunque pudiera parecer muy alto, supone aproximadamente un tercio del presupuesto anual para desarrollo de IBM.

Para los japoneses, este dinero supone un esfuerzo. Sin embargo, la tecnología es la base de sus exportaciones, y éstas son vitales para ellos, significan su supervivencia. El país está superpoblado, sus necesidades petrolíferas sólo se cubren en un 1 por 100 con crueros del país, y considerando los restantes productos energéticos, los nacionales sólo llegan a suministrar el 15 por 100 de las necesidades totales del país. Además, el japonés, por educación, es un hombre educado y muy disciplinado. Su empresa es su segunda familia, y, por lo general, trabaja con entusiasmo para que suba. A su vez, las empresas son paternalistas. El orgullo japonés no soporta el «sambenito» de copiones, con que muchos los designan. Además, el escándalo de Hitachi y Mitsubishi en el que estas dos empresas japonesas fueron acusadas de espionaje industrial por la IBM, supuso un enorme salto atrás en la consideración mundial. (Realmente, el espionaje industrial es un ejercicio que muchas empresas potentes llevan a cabo con cierta asiduidad. No es en absoluto un pecado japonés). Sin embargo, los que fueron descubiertos fueron los japoneses.

La historia, sucintamente narrada, comienza en el verano de 1982, cuando varios directivos japoneses de Hitachi y Mitsubishi se vieron implicados en un escándalo de intento de robo de secretos a IBM. El caso tuvo más importancia de la prevista, ya que el gobierno norteamericano llevaba en ese momento unas negociaciones con el Japón, y ese hecho fue un arma con la que obtuvo ventajas sustanciales. El gobierno y el pueblo japonés quedaron ingenuamente avergonzados; estaban en entredicho.

El asunto, probablemente, fuera un problema de «compatibilidad IBM». Los japoneses tienen tecnología suficiente para diseñar y fabricar sus propios circuitos y elementos sin necesidad de acudir a las soluciones de otros. Lo más probable es que, al fabricar ordenadores compatibles IBM, intentaran llevar aún más lejos dicha «compatibilidad».

Volviendo al presupuesto anterior, a pesar de ser una suma considerable, no es en absoluto un derroche para el pueblo japonés, que necesita salir a otros mercados obligadamente.

¿Qué ocurre mientras tanto en Europa? Los franceses, por ejemplo, han aumentado su presupuesto para investigación, pasando del 6 al 8 por 100 (por encima de la inflación). El gobierno francés desde hace bastante tiempo está concienciado de que es muy importante estar a la cabeza en tecnología, y que las grandes sumas dedicadas a investigación son rentables, a largo plazo, desde luego. La ciencia más favorecida es la electrónica. El plan de desarrollo de estas tecnologías fue comenzado por el presidente Giscard, aunque Mitterrand lo ha tomado como propio. En la reunión de Jefes de Estado, celebrada en Versalles en 1982, se declaró partidario de impulsar la investigación, sobre todo en Informática y Telecomunicaciones. Frente a otras voces que propugnaban sobriedad, Mitterrand defendía la salida de la crisis invirtiendo en tecnologías punta.

En Francia tiene su sede el Centro Mundial de Tecnología Informática y Recursos Humanos. Este centro fue en cierto modo concebido por Jean Jacques Servan-Schreiber. Tiene un presupuesto muy alto (9 millones de dólares), y por él han pasado hombres insignes. Existe, sin embargo, un cierto desasosiego entre sus dirigentes, y muchos convienen en que la intervención estatal es excesiva, y que existen otros intereses ajenos a la entidad. Seymour Papet, por ejemplo, dimitió como director científico quejándose de que existían demasiadas inferencias políticas que le impedían desarrollar su labor.

La verdad es que existen muchos organismos franceses cuyos presupuestos para investigación son más que exigüos, lo que provoca ciertos recelos y envidias.

Los alemanes, a su vez, tienen una actitud mucho más conservadora que los franceses. Nixdorf envió observadores a la Conferencia de Quinta Generación de Tokio, en 1981. Todos ellos opinaron favorablemente sobre el proyecto. Sin embargo, a nivel estatal no parecen demasiado apresurados en la toma de medidas.

Los británicos dedican cada vez menos a investigación. Además, los comités que deciden sobre las ayudas estatales están dominados por físicos, siendo, por tanto, la Física la ciencia que más ayuda recibe. Sin embargo, a nivel empresarial, la Gran Bretaña siempre ha destacado en sus inversiones en investigación. No ocurre lo mismo ahora. El dinero invertido por las empresas ha descendido desde 1964

hasta 1975, y la curva hasta nuestros días sigue la misma tendencia. Sin embargo, los porcentajes dedicados a investigación tienen cuantías similares a las dedicadas por los japoneses. La diferencia está en que los británicos están acostumbrados a estar en la avanzada de los países industrializados, y el Japón, desde luego, muy industrializado, está conociendo una era de crecimiento sin parangón, lo que supone una enorme ilusión y orgullo entre todos los que cooperan para que este crecimiento continúe, e incluso se impulse. El Comité Alvey fue creado por el gobierno británico para asesorar en materia de investigación. En 1982 publicó un informe en el que admitía que su creación se debía al desafío japonés, y recomendaba hacer un esfuerzo en investigación, sobre todo en Informática y comunicaciones, para no quedar estancados en los años venideros.

Los norteamericanos, mientras tanto, comenzaron lentamente a despertar de su asombro ante el desafío japonés. El primero en interesarse fue el profesor Feigenbaum, a quien los japoneses habían visitado antes de la conferencia de octubre de 1981 para ir tomando contactos. El profesor Arvind, de Stanford, volvió de la Conferencia y redactó otro informe que estaba de acuerdo casi en todo con las opiniones e informes de Feigenbaum. Preocupado, se reunió con el director del Laboratorio de Informática del Massachusetts Institute of Technology, Michael Dertouzos, y algunas otras cabezas pensantes de primera línea en el campo de la Informática. Dertouzos y su departamento habían llegado a conclusiones parecidas a las de los japoneses, por lo que quedaron sorprendidos y al mismo tiempo convencidos de que el proyecto era viable.

Decidieron, por tanto, ponerse en acción, y escribieron a las direcciones de las más importantes firmas norteamericanas (IBM, Honeywell, Intel, Data General, Digital Equipment, etc.), para comunicarles sus enormes preocupaciones. También escribieron a la Agencia de Proyectos Avanzados de Investigación del Departamento de Defensa. Las respuestas fueron rápidas por lo general, llegando a interesarse en el tema el subsecretario de Defensa, Richard de Lauer. Tras una reunión en Orlando (Florida), que estuvo marcada por un afán común de cooperación, así como una sensación de que era necesario tomar medidas inmediatamente, tanto por parte del gobierno como de las empresas. El gobierno debería ofrecer créditos a corto plazo para

TEMAS MONOGRAFICOS DE VANGUARDIA

invertir en investigación, y se debería crear un consorcio con todos los Institutos no comerciales, que funcionara como regulador de todos los esfuerzos, evitando duplicidades.

Sin embargo, a pesar de todo lo expuesto, este movimiento no era en absoluto mayoritario. Muchos investigadores de renombre no creían en absoluto en el proyecto japonés, pensando que se trataba de un montaje publicitario. Dado el carácter nipón, al que no gusta concretar las cosas, los planes parecían estar llenos de vaguedades. Y además, los objetivos a cumplir eran ambiciosísimos. ¿Cómo alcanzar esos objetivos tan altos con unos planes tan poco concretos? Es evidente que el carácter japonés puede tener inconvenientes, pero desde luego tiene ventajas. Y hasta ahora han demostrado que pueden avanzar e incluso aventajar tecnologías de vanguardia. Por ejemplo, Hewlett Packard ha suscrito un contrato para fabricar, bajo licencia de Hitachi, pastillas de memoria japonesas. Efectivamente, los japoneses han desarrollado pastillas de memoria RAM de 64 K) baratas y seguras. Además, están ya avanzados en el desarrollo de las pastillas de 256 K. En cuanto al vídeo, aunque sea un invento norteamericano,

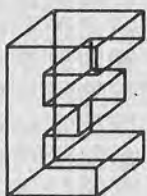
los que lo han desarrollado y mejorado han sido desde luego los japoneses, que además, han tenido visión de futuro como para comprender que podía ser un producto de consumo de primer orden.

Sin embargo, el orgullo de «pionero americano» se resiste a creer en que los pequeños e inteligentes japoneses puedan alcanzar cotas tecnológicas tan altas como su nación.

En la Europa comunitaria también existe otro proyecto, creado con cierta aceleración, como respuesta al desafío japonés. El proyecto ESPRIT. A finales de los años setenta, Europa conoció un enorme auge tecnológico, que ha ido decayendo año a año, hasta llegar a la situación actual (5.000 millones de dólares de superávit en 1970, 12.000 millones de déficit en 1984). Es necesario cambiar los esquemas y trabajar sin pausa para retomar la situación anterior.

En este camino, en Baviera, en 1984 se fundó un centro financiado por varias Compañías de las más importantes europeas dedicadas a la Informática: la International Computers, de Gran Bretaña; la Bull, de Francia, y la enorme Siemens, de Alemania.

LOS LENGUAJES DE LA QUINTA GENERACION



N la Sexta Conferencia Internacional de Software, celebrada en Tokio (1982), se reunieron mil ingenieros japoneses, cien representantes de los Estados Unidos y Canadá y cincuenta europeos. También se encontra-

ban representantes del ICOT (comité fundado por el Ministerio de Defensa japonés).

En lo que se refiere al Logical, se creó un grupo de sistemas para diseñar un esquema de producción óptima, al que habrán de ajustarse los sistemas de proceso de la información. Este sistema conocerá los detalles de lo que se va a producir, e incluirá subsistemas capaces de desarrollar, a partir de la información escrita, una información final, verificando la exactitud y simulando las operaciones implicadas. El sistema de soporte incluirá otro sistema de programación inteligente para el tratamiento de programas, un sistema de diseño de una base inteligente para gestionar la base de conocimientos y un sistema inteligente para el diseño de microcircuitos VLSI. También dispondrá de funciones muy sofisticadas. Podrá transferir programas y bases de datos de otros sistemas al sistema objeto, para diagnóstico inteligente de averías y reparaciones (mantenimiento y reparaciones).

El Logical básico podrá llamar a tres bases de conocimientos universales integradas dentro del sistema de gestión de la base de conocimientos. Las salidas serán en un lenguaje ordinario.

En la quinta generación podrán utilizarse tres tipos de lenguajes:

- Lenguaje natural, verbal y gráfico.
- Lenguaje del núcleo del sistema

operativo de la quinta generación, es decir, el utilizado por el procesador.

— Lenguaje interpretativo: lenguaje de alto nivel para el sistema de gestión.

Si se desea programar el ordenador, sólo se utilizará el lenguaje de programación del núcleo. Este lenguaje irá directamente a los procesadores de bases de conocimientos y sistemas para la resolución del problema.

El lenguaje seleccionado por el comité para la quinta generación japonés es el PROLOG.

Este lenguaje está estructurado a base de cláusulas que son declaraciones lógicas restringidas, para que al ejecutarlo se pueda llegar a deducciones lógicas. Sin embargo, también se podrá interactuar con la máquina en lenguaje conversacional.

El PROLOG se utiliza junto con una forma básica de Lógica matemática, la resolución, que resuelve problemas utilizando inferencias lógicas.

En PROLOG, la unidad básica es una cláusula de Horn, que puede ser una declaración directa o aseveración; o una implicación.

Hay toda una gama de opiniones controvertidas respecto al PROLOG, considerado como lenguaje de la quinta generación. Algunos opinan que es el lenguaje ideal para la inteligencia artificial, y algunos otros opinan que es el talón de Aquiles del proyecto japonés.

Otro de los lenguajes susceptible de ser utilizado es el LISP (List Programming). Este lenguaje maneja listas. Realmente se trata de un metalenguaje (un lenguaje fuente de otros lenguajes).

Existe la teoría de que los japoneses no pueden producir software porque tienen dificultades con su propio lenguaje. El japonés es

LOS LENGUAJES DE LA QUINTA GENERACION

un lenguaje relacionado con el turco o el mōngol, pero que adoptó la escritura de caracteres y estructura china, posiblemente por proximidad. A los occidentales la forma de enfocar los problemas sin ir directamente a la cuestión, formulando lentamente los argumentos, nos resulta poco clara. Preferimos la rapidez, la exposición clara y contundente del problema, frente a dar ejemplos de los que deban extraerse conclusiones.

Los japoneses llevan un atraso real en programación de unos tres años (muchos afirman que llevan un atraso mayor, pero no es cierto). Además, aunque no estén a la cabeza en los lenguajes utilizados en nuestros días eso no significa que no nos aventajen utilizando lenguajes nuevos, o innovados, y pueden dejar a los occidentales verdaderamente «en la cuneta».

Otro de los motivos aducidos por los detractores del proyecto es que en los procesos deductivos de la máquina pocos profesionales cederían totalmente el control al ordenador en un proceso de demostración de teoremas que lleve a cabo búsquedas masivas, sin demasiado control sobre el tiempo invertido.

El futuro de la Informática está en el tratamiento del saber. Los japoneses se han dado cuenta, y han reaccionado a tiempo. El Japón es rico, porque las exportaciones sobrepasan las importaciones. Además, las exportaciones se ven enriquecidas por el valor del potencial humano japonés (capacidad, inteligencia, tesón). El valor de las materias importadas para la industria informática es casi nulo.

VOCABULARIO DE INFORMATICA

Chain. (Véase **Cadena.**)

Check. (Véase **Verificación.**)

Checksum. Control que se utiliza para evitar posibles errores en la grabación de registros. Su valor es la suma binaria de todos los caracteres del registro. En contraposición con el control de paridad.

Chip. Microcircuito electrónico, compuesto por una oblea de silicio sobre la que se encuentra un circuito electrónico completo, con transistores, diodos, etc. Modernamente se pueden crear circuitos complejísimos, que incluyen decenas de miles de componentes electrónicos.

DAC. (Digital Analogic Converter). Véase **Conversor Analógico Digital.**

Data processing. (Véase **Proceso de Datos.**)

Dato. Representación de hechos, conceptos o instrucciones de forma adecuada para su interpretación, proceso o comunicación por medios humanos o automáticos.

Cualquier carácter o dígito que se encuentre en el ordenador o en su memoria para su proceso.

Dato, nombre del. Identificador que asigna un nombre a un dato de forma inequívoca.

Datos, base de. Conjunto de datos organizados de forma estructurada. Por ejemplo, una línea de una factura puede considerarse un elemento, una factura completa un registro, y el conjunto de registros constituye el fichero. El conjunto de ficheros forma la

librería, y el conjunto de librerías constituye el banco de datos.

Datos, banco de. Banco de datos en el que las informaciones están almacenadas con una estructura independiente de los programas que la utilizan. Su flexibilidad, independencia de estructura y control de fiabilidad (no redundancia de datos) permiten el almacenamiento cómodo de grandes cantidades de datos.

Datos, bus de. Canal paralelo (bus) dedicado exclusivamente a transmisión de datos. Normalmente dispone de tantas líneas como bits tiene el carácter (byte o palabra) a transmitir.

Datos, carácter de escape de un enlace de. En comunicaciones, carácter de control que se utiliza junto con uno o más caracteres añadidos para formar una secuencia de escape y obtener en operaciones de control de comunicaciones adicionales. Para estas secuencias de escape sólo se utilizan caracteres gráficos y caracteres de control.

Datos, código de. Conjunto de caracteres estructurados que se utilizan para representar los elementos o grupos de datos. Por ejemplo, los códigos 1, 2, 12 se suelen utilizar para representar los meses del año.

Datos, diagrama de. Representa la trayectoria que siguen los datos en la resolución de un problema. Define las fases principales del proceso y los medios utilizados en la resolución.

VOCABULARIO DE INFORMATICA

Datos, dispositivo destino de. En comunicaciones dispositivo capaz de aceptar datos de un dispositivo de transmisión. También puede comprobar las señales y emitir otras de control.

Datos, dispositivo fuente de. En comunicaciones, dispositivo capaz de originar señales con los datos, para que sean recibidas por otros dispositivos. También puede aceptar señales de control (errores).

Datos, grupo de. Nombre de una clase o categoría de datos, basado en las relaciones naturales o asignadas y que puede utilizarse para indicar un conjunto de elementos de dato. Por ejemplo, "martes" constituye un elemento del grupo "días de la semana".

Datos, jerarquía de los. Estructura formada por conjuntos y subconjuntos de datos, de forma que cada subconjunto de un conjunto dado sea de un rango inferior al de los datos del conjunto.

Datos, registro de (logging). Anotación de datos referentes a acontecimientos que tienen lugar secuencialmente en el tiempo.

Datos, terminal de. Puede tratarse de un dispositivo emisor o receptor de datos. Véase **Datos, dispositivo destino de**, y **Datos, dispositivo fuente de**.

Datos, velocidad de. En comunicaciones, capacidad de transmisión de los datos de un canal o conjunto de canales. La velocidad de transmisión de los datos viene indicada en bits por segundo.

DBA (Digital Differential Analyzer). (Véase **Digital, analizador diferencial**.)

Debug. (Véase **Depuración**.)

Decimal. Sistema de numeración en el que la base es diez. Perteneciente o característico de una selección en la que se puede optar entre diez posibilidades.

Decimal, dígito. Uno de los caracteres de la notación decimal (del 0 al 9).

Decimal, punto. Punto que separa la parte entera de los decimales, sin una representación decimal.

Decisión. Determinación de la futura acción a tomar. En todos los lenguajes de programación son básicas las instrucciones de que se dispone para la toma de decisiones en los procedimientos.

Decisión, instrucción de. Instrucción que selecciona una de las ramas del programa. Instrucción de salto condicional.

Decisión, tabla de. Tabla que contiene todas las contingencias que deben considerarse en la descripción del problema, junto con las acciones a tomar. Se utilizan con frecuencia en la descripción y documentación de problemas, frente a los diagramas de flujo.

Decodificar. Aplicar un conjunto de reglas inequívocas disponibles para volver a obtener los datos en su representación previa. Es el inverso de **Codificar**.

Decodificador. Dispositivo que decodifica; matriz de elementos lógicos que selecciona uno o más canales de salida de acuerdo con la combinación de señales de entrada de que se dispone.

Dedicado, ordenador. Ordenador especializado en una tarea a la que se dedica exclusivamente.

DEL. Apócope de DELETE. Representación del carácter de borrado.

Delay. (Véase **Retardo**.)

Delay line. (Véase **Línea de retardo**.)

Delimitador. Indicador que separa los datos organizándolos como elementos. Sinónimo de separador.

Densidad. Medida de capacidad física de almacenamiento de los datos.

Depurador (debugger). Programa especializado en la búsqueda sistemática y corrección de errores.

Depurar. Eliminar errores de un programa.

Desbordamiento. Sobrepassamiento de la capacidad de almacenamiento de un registro, un archivo o un dispositivo.

